Πανεπιστήμιο Δυτικής Μακεδονίας
Τμήμα Μηχανικών Πληροφορικής και Τηλεπικοινωνιών

Διπλωματική Εργασία

Επεξεργαστής, μεταγλωττιστής και προσομοιωτής HDL στο σύννεφο και διαδικτυακή εφαρμογή για την σχεδίαση κυματομορφών σημάτων από HDL προσομοίωση

A cloud based HDL editor, compiler and simulator and a web based signal waveform plot application for HDL simulation

Λυμπερίδης Ευστάθιος
Επιβλέπων Καθηγητής: **Δασυγένης Μηνάς**

Εργαστήριο Ψηφιακών Συστημάτων και Αρχιτεκτονικής Υπολογιστών

Κοζάνη, 2016

Περίληψη

Στην σύγχρονη εποχή συναντάμε την ανάπτυξη και χρήση διαδικτυακών εφαρμογών όλο και περισσότερο. Καθώς οι δυνατότητες εξυπηρετητών συνεχώς αναπτύσσονται και οι μέγιστες ταχύτητες διαδικτύου πιέζονται όλο και περισσότερο σε νέα όρια, βλέπουμε την σταδιακή αλλά και σταθερή μετάβαση από τοπικές εφαρμογές σε εφαρμογές διαδικτύου. Η μετάβαση αυτή γίνεται ιδιαίτερα επιτακτική σε τομείς όπου η χρήση τοπικών προγραμμάτων είναι πολύπλοκη, χρονοβόρα και απαιτεί σημαντική εξοικείωση με αυτά. Ένας από αυτούς τους τομείς είναι και η συγγραφή και εξομοίωση HDL (hardware description language, γλώσσα περιγραφής υλικού) κώδικα.

Στην παρούσα διπλωματική εργασία αναλάβαμε την σχεδίαση και υλοποίηση μίας πλήρους διαδικτυακής εφαρμογής που προσφέρει ένα εύχρηστο και αποτελεσματικό περιβάλλον για την ανάπτυξη και εξομοίωση HDL προγραμμάτων. Η εφαρμογή που παρουσιάζεται περιλαμβάνει, μεταξύ άλλων, ολοκληρωμένο σύστημα χρηστών αλλά και την δυνατότητα περιορισμένης πρόσβασης για επισκέπτες, κειμενογράφο με επισημάνσεις συντακτικού και άλλες διευκολύνσεις στην συγγραφή κώδικα, την διατήρηση διαφορετικών έργων με πολλαπλούς συντάκτες και εσωτερική διαχείριση αρχείων, την χρήση και διανομή εξαρτημάτων ή βιβλιοθηκών με άλλους χρήστες καθώς και σύστημα απεικόνισης κυματομορφών με δυνατότητες λεπτομερούς ανάλυσής τους από τον χρήστη. Η εφαρμογή που αναπτύχθηκε προσφέρει πλέον όχι μόνο τις απαραίτητες λειτουργίες για την συγγραφή HDL προγραμμάτων αλλά και λειτουργίες που ενισχύουν και διευκολύνουν την όλη διαδικασία.

Κατά την ανάπτυξη της εργασίας δόθηκε προσοχή στην κατασκευή ενός ολοκληρωμένου εργαλείου που καλύπτει την ανάγκες ενός προγραμματιστή. Ως αποτέλεσμα, προσφέρουμε ταυτόχρονα ένα περιβάλλον όπου έμπειροι προγραμματιστές θα βρουν οικείο και κοντά στα εργαλεία με τα οποία έχουν ήδη εξοικειωθεί ενώ νέοι προγραμματιστές θα μπορέσουν να χρησιμοποιήσουν χωρίς περαιτέρω εκπαίδευση. Μεταφέροντας την διαδικασία δημιουργίας και προσομοίωσης HDL προγραμμάτων σε ένα διαδικτυακό περιβάλλον, βλέπουμε ήδη όχι μόνο βελτίωση αλλά και επέκταση αυτής της διαδικασίας. Θέτουμε πλέον σημαντικές βάσεις για την είσοδο της συλλογικότητας σε αυτή την διαδικασία όπου έχουμε όχι μόνο την ανταλλαγή ιδεών και κώδικα αλλά και την ταυτόχρονη συγγραφή του προγράμματος από μια ομάδα προγραμματιστών.

Λέξεις Κλειδιά : Εφαρμογή στο σύννεφο, Ιστότοπος, Προγραμματισμός HDL, Προσομοίωση κυματομορφών, PHP, JavaScript, MySQL

Abstract

In modern times we meet the development and deployment of web applications increasingly often. As servers' capabilities are constantly developing and maximum internet speeds ask more and more pressure for new boundaries, we see the gradual but steady transfer of local applications to web applications. This transfer is particularly pressing on areas where the use of local programs is complex, time consuming and requires considerable familiarity with them. One of these areas is the development and simulation of HDL (hardware description language) code.

In this thesis project, we undertook the design and implementation of a complete web application that offers a friendly and efficient environment to develop and simulate HDL programs. The presented application includes, inter alia, an integrated user system with the ability of limited access for visitors, text editor with syntax highlights and other facilities in writing code, the retention of different projects with multiple editors and internal file management, the use and distribution of components or libraries with other users and a waveform viewer system, featuring functions for detailed analysis by the user. The application developed offers not only the necessary functions for writing HDL programs, but also operations that enhance and facilitate the whole process.

During the project's development, attention was paid for the construction of an integrated tool that covers the needs of a developer. As a result, we offer simultaneously an environment where experienced developers will find familiar and close to the tools they are already acclimatized with, while new developers will be able to use without further training. Transporting the creation and simulation process of HDL programs in an online environment, we are already seeing not only the improvement but also an extension of this process. We set important bases for the entrance of collectivity in this process where there is not only the sharing of ideas and code, but also the simultaneous development by a team of programmers.

Keywords: Cloud application, Website, HDL programming, Waveforms simulation, PHP, JavaScript, MySQL

Δήλωση Πνευματικών Δικαιωμάτων

Δηλώνω ρητά ότι, σύμφωνα με το άρθρο 8 του Ν. 1599/1986 και τα άρθρα 2,4,6 παρ. 3 του Ν. 1256/1982, η παρούσα Διπλωματική Εργασία με τίτλο "Επεξεργαστής, μεταγλωττιστής και προσομοιωτής HDL στο σύννεφο και διαδικτυακή εφαρμογή για την σχεδίαση κυματομορφών σημάτων από HDL προσομοίωση" καθώς και τα ηλεκτρονικά αρχεία και πηγαίοι κώδικες που αναπτύχθηκαν ή τροποποιήθηκαν στα πλαίσια αυτής της εργασίας και αναφέρονται ρητώς μέσα στο κείμενο που συνοδεύουν, και η οποία έχει εκπονηθεί στο Τμήμα Μηχανικών Πληροφορικής και Τηλεπικοινωνιών του Πανεπιστημίου Δυτικής Μακεδονίας, υπό την επίβλεψη του μέλους του Τμήματος κ. Δασυγένη Μηνά, αποτελεί αποκλειστικά προϊόν προσωπικής εργασίας και δεν προσβάλλει κάθε μορφής πνευματικά δικαιώματα τρίτων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον. Τα σημεία όπου έχω χρησιμοποιήσει ιδέες, κείμενο, αρχεία ή / και πηγές άλλων συγγραφέων, αναφέρονται ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή.

Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα. Ερωτήματα που αφορούν τη χρήση της εργασίας για κερδοσκοπικό σκοπό πρέπει να απευθύνονται προς τον συγγραφέα. Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν τον συγγραφέα και μόνο.

Περιεχόμενα

Περίληψη *4*

Abstract *6*

Δήλωση Πνευματικών Δικαιωμάτων *7*

Κατάλογος Εικόνων *10*

Κατάλογος Πινάκων *10*

Κατάλογος Κωδικών *10*

Κεφάλαιο 1 - Εισαγωγή *12*

1.1 Κίνητρα και Στόχοι Υλοποίησης **13**

1.2 Σχετικά Εργαλεία **14**

1.3 Διάρθρωση Βιβλίου **16**

Κεφάλαιο 2 - Θεωρητικό Υπόβαθρο *18*

2.1 Βασικά Εργαλεία Προγραμματισμού Διαδικτύου **18**

2.1.1 HTML και CSS 19

2.1.2 JavaScript 20

2.1.3 PHP 21

2.1.4 MySQL 22

2.1.5 Πρωτόκολλο HTTP 23

2.2 Εξωτερικά Εργαλεία που Χρησιμοποιήθηκαν **24**

2.2.1 Πλαίσιο Bootstrap 24

2.2.2 Βιβλιοθήκη jQuery 25

2.2.3 Κειμενογράφος Ace 25

2.2.4 GHDL 26

2.2 Σύνοψη Κεφαλαίου **26**

Κεφάλαιο 3 - Ανάπτυξη της Εφαρμογής *28*

3.1 Διάρθρωση της Βάσης Δεδομένων **28**

3.1.1 Περιγραφή του Πίνακα users 30

3.1.2 Περιγραφή του Πίνακα user_activation 32

3.1.2 Περιγραφή του Πίνακα projects 32

3.1.3 Περιγραφή του Πίνακα project_files 33

3.1.4 Περιγραφή του Πίνακα sid_files 34

3.1.5 Περιγραφή του Πίνακα projects_editors 35

3.1.6 Περιγραφή του Πίνακα libraries 36

3.1.7 Περιγραφή του Πίνακα library_updates 37

3.2 Κατάλογοι Αρχείων **38**

3.2.1 Αρχεία Χρηστών και Βιβλιοθηκών 38

3.2.2 Αρχεία Επισκεπτών 39

3.2.3 Αρχεία Εργασιών και Αναφοράς Κατάστασης 40

3.3 Ανάπτυξη και Υλοποίηση της Εφαρμογής **40**

3.3.1 Αρχικοποίηση της Εφαρμογής κατά την Φόρτωση 41

3.3.2 Ανάλυση των Διαχειριστών Αλληλεπίδρασης 44

3.3.3 Ανάλυση των Κλάσεων που Χρησιμοποιήθηκαν 46

3.3.4 Υλοποίηση και Χρήση του Θέματος Εμφάνισης ... 53
3.3.5 Τεχνικές Υλοποίησης των Θεμάτων Ασφάλειας ... 57

3.4 Υλοποίηση του Εργαλείου Σχεδίασης Κυματομορφών ... 61
3.4.1 Εξαγωγή και Μεταφορά Δεδομένων ... 61
3.4.2 Ανάλυση Τεχνικής Σχεδίασης Κυματομορφών ... 63
3.4.3 Ανάλυση Υλοποίησης Βασικών Λειτουργιών ... 67

3.5 Οδηγίες Εγκατάστασης ... 72

3.6 Σύνοψη Κεφαλαίου ... 74

Κεφάλαιο 4 - Περιγραφή Λειτουργιών ... 76

4.1 Λειτουργίες Έργων ... 76
4.1.1 Δημιουργία, Διαγραφή και Επεξεργασία Έργου ... 77
4.1.2 Δημιουργία, Διαγραφή, Μεταφόρτωση και Επεξεργασία Αρχείων ... 79
4.1.3 Μεταγλώττιση και Προσομοίωση ... 81
4.1.4 Λειτουργίες του Εργαλείου Σχεδίασης Κυματομορφών ... 82
4.1.5 Δυνατότητες Διαμοίρασης Έργων ... 84

4.2 Λειτουργίες Εξαρτημάτων ... 85
4.2.1 Ανάρτηση και Λήψη Εξαρτημάτων ... 85
4.2.2 Διαδικασία Ενημερώσεων Εξαρτημάτων ... 87
4.2.3 Προβολή και Βαθμονόμηση Εξαρτημάτων ... 88

4.3 Επίπεδα Χρηστών ... 89
4.3.1 Δυνατότητες Επισκεπτών ... 89
4.3.2 Δυνατότητες Χρηστών ... 90
4.3.3 Δυνατότητες Διαχειριστών ... 91

4.4 Σύνοψη Κεφαλαίου ... 92

Κεφάλαιο 5 - Επίλογος ... 94

5.1 Συμπεράσματα ... 94
5.1.1 Ανάλυση Μετρικών Κώδικα ... 95
5.1.2 Ανάλυση S.W.O.T. ... 97

5.2 Μελλοντικές Επεκτάσεις ... 99

Βιβλιογραφία ... 102

Κατάλογος Εικόνων

Εικονα 1: Ένθετος κωδικας PHP σε κωδικα HTML 22
Εικονα 2: Η δομη της βασης δεδομενων 30
Εικονα 3: Η δομη του πινακα users 31
Εικονα 4: Η δομη του πινακα user_activation 32
Εικονα 5: Η δομη του πινακα projects 33
Εικονα 6: Η δομη του πινακα project_files 34
Εικονα 7: Η δομη του πινακα sid_files 35
Εικονα 8: Η δομη του πινακα projects_editors 35
Εικονα 9: Η δομη του πινακα libraries 36
Εικονα 10: Η δομη του πινακα library_updates 37
Εικονα 11: Οθονη δημιουργιας νεου εργου 77
Εικονα 12: Οθονη ενος εργου για χρηστη με δικαιωματα ιδιοκτητη σε αυτο 79
Εικονα 13: Οθονη επεξεργασιας αρχειου 80
Εικονα 14: Οθονη του εργαλειου σχεδιασης κυματομορφων 83
Εικονα 15: Αρχικη οθονη συνδεμενου χρηστη 84
Εικονα 16: Οθονη διαχειρισης εξαρτηματων 86
Εικονα 17: Οθονη επεξεργασιας προτεινομενης ενημερωσης 87
Εικονα 18: Οθονη προβολης ολων των εξαρτηματων που εχουν εγκριθει 88
Εικονα 19: Αρχικη οθονη επισκεπτη 90
Εικονα 20: Οθονη επιλογων λογαριασμου χρηστη 91
Εικονα 21: Οθονη διαχειρισης χρηστων 92
Εικονα 22: Επιβεβαιωση σωστης σχεδιας της εφαρμογης για το προτυπο HTML5 απο το εργαλειο που προσφερει η W3C 95

Κατάλογος Πινάκων

Πινακας 1: Μετρικες αρχειων και κωδικα, αποτελεσματα του προγραμματος CLOC 96
Πινακας 2: Βασικες μετρικες κωδικα PHP, αποτελεσματα του προγραμματος PHPLOC 96
Πινακας 3: Κυκλωματικη πολυπλοκοτητα κωδικα PHP, αποτελεσματα του προγραμματος PHPLOC 97
Πινακας 4: Αναλυση S.W.O.T. της εφαρμογης 98

Κατάλογος Κωδικών

Κωδικας 1: loader.php - Έλεγχος και δημιουργια συνεδριας 41
Κωδικας 2: loader.php - Αρχικοποιηση καταλογων 41
Κωδικας 3: loader.php - Αρχικοποιηση αντικειμενων 42
Κωδικας 4: loader.php - Αρχικοποιηση συνεδριας και αντικειμενου χρηστη 42
Κωδικας 5: loader.php - Κλιση αρχειων ασφαλειας και αλληλεπιδρασης 42
Κωδικας 6: index.php - Κλιση του loader.php και δρομολογηση 43
Κωδικας 7: index.php -Κλιση αρχειων θεματος εμφανισης και εμφανιση μηνυματων 44
Κωδικας 8: post_handler.php - Γενικη δομη του αρχειου 45
Κωδικας 9: ajax_handler.php - Γενικη δομη του αρχειου 46
Κωδικας 10: Database.php - Η μεθοδος κατασκευης της κλασης Database 47
Κωδικας 11: Database.php - Η μεθοδος επιβεβαιωσης χρηστη 48
Κωδικας 12:Messages.php - Οι πινακες μηνυματων και κειμενου της κλασης Messages 48
Κωδικας 13: Messages.php - Μεθοδος εμφανισης μηνυματων 49

ΚΩΔΙΚΑΣ 14: USER.PHP - ΟΡΙΣΜΟΣ ΜΕΤΑΒΛΗΤΩΝ ΚΑΙ Η ΜΕΘΟΔΟΣ ΚΑΤΑΣΚΕΥΗΣ ΤΗΣ ΚΛΑΣΗΣ USER 50
ΚΩΔΙΚΑΣ 15: USER.PHP - Η ΜΕΘΟΔΟΣ ΕΠΑΛΗΘΕΥΣΗΣ ΙΔΙΟΚΤΗΣΙΑΣ ΕΝΟΣ ΕΡΓΟΥ ΑΠΟ ΤΟΝ ΧΡΗΣΤΗ 50
ΚΩΔΙΚΑΣ 16: USER.PHP - Η ΜΕΘΟΔΟΣ ΕΠΑΛΗΘΕΥΣΗΣ ΔΙΚΑΙΩΜΑΤΩΝ ΣΥΓΓΡΑΦΕΑ..... 51
ΚΩΔΙΚΑΣ 17: GENERAL.PHP - ΟΙ ΜΕΘΟΔΟΙ EXTRACT_FILE ΚΑΙ SEND_EMAIL 52
ΚΩΔΙΚΑΣ 18: GENERAL.PHP - Η ΜΕΘΟΔΟΣ CREATE_SHORT_CODE..... 52
ΚΩΔΙΚΑΣ 19: HEADER.PHP - Ο ΚΩΔΙΚΑΣ ΤΟΥ ΑΡΧΕΙΟΥ 53
ΚΩΔΙΚΑΣ 20: NAV-BAR.PHP - Η ΓΕΝΙΚΗ ΔΟΜΗ ΤΗΣ ΜΠΑΡΑΣ ΕΡΓΑΛΕΙΩΝ ΜΕ ΤΙΣ ΚΛΑΣΕΙΣ ΠΟΥ ΠΡΟΣΦΕΡΕΙ ΤΟ ΠΛΑΙΣΙΟ BOOTSTRAP 54
ΚΩΔΙΚΑΣ 21: FOOTER.PHP - Ο ΚΩΔΙΚΑΣ ΤΟΥ ΑΡΧΕΙΟΥ 55
ΚΩΔΙΚΑΣ 22: FUNCTIONS.JS - Η ΓΕΝΙΚΗ ΔΟΜΗ ΤΟΥ ΑΡΧΕΙΟΥ 56
ΚΩΔΙΚΑΣ 23: SECURITY.PHP - ΑΡΧΙΚΟΠΟΙΗΣΗ ΚΑΙ ΕΛΕΓΧΟΣ ΓΕΝΙΚΩΝ ΔΕΔΟΜΕΝΩΝ ΣΥΜΒΟΛΟΣΕΙΡΩΝ 57
ΚΩΔΙΚΑΣ 24: SECURITY.PHP - ΈΛΕΓΧΟΣ ΑΣΦΑΛΕΙΑΣ ΓΙΑ ΤΑ ΔΕΔΟΜΕΝΑ SELECTED_IDS ΚΑΙ EMAIL_EDIT 58
ΚΩΔΙΚΑΣ 25: SECURITY.PHP - ΑΝΤΙΜΕΤΩΠΙΣΗ ΠΙΘΑΝΗΣ ΠΑΡΑΒΙΑΣΗΣ ΑΣΦΑΛΕΙΑΣ 58
ΚΩΔΙΚΑΣ 26: DATABASE.PHP - Η ΜΕΘΟΔΟΣ ΓΕΝΙΚΗΣ ΑΝΑΖΗΤΗΣΗΣ ΧΡΗΣΤΗ ΑΠΟ ΤΗΝ ΒΑΣΗ ΔΕΔΟΜΕΝΩΝ 59
ΚΩΔΙΚΑΣ 27: INDEX.PHP - ΚΩΔΙΚΑΣ ΤΟΥ ΑΡΧΕΙΟ ΑΝΑΚΑΤΕΥΘΥΝΣΗΣ..... 60
ΚΩΔΙΚΑΣ 28: ΚΩΔΙΚΑΣ ΑΝΑΚΑΤΕΥΘΥΝΣΗ ΠΟΥ ΒΡΙΣΚΕΤΑΙ ΣΕ ΚΑΘΕ ΑΡΧΕΙΟ ΣΕΛΙΔΑΣ..... 60
ΚΩΔΙΚΑΣ 29: AJAX_HANDLER.PHP - ΚΟΜΜΑΤΙ ΚΩΔΙΚΑ ΕΞΑΓΩΓΗΣ ΔΕΔΟΜΕΝΩΝ ΑΠΟ ΤΟ VCD ΑΡΧΕΙΟ..... 62
ΚΩΔΙΚΑΣ 30: AJAX_HANDLER.PHP - Ο ΕΛΕΓΧΟΣ ΚΑΙ ΔΗΜΙΟΥΡΓΙΑ ΤΟΥ ΕΝΔΙΑΜΕΣΟΥ ΑΡΧΕΙΟΥ JSON 62
ΚΩΔΙΚΑΣ 31: WAVEFORMS_VIEWER.JS - ΑΡΧΙΚΟΠΟΙΗΣΕΙΣ ΚΑΙ ΥΠΟΛΟΓΙΣΜΟΙ ΜΕΤΑΒΛΗΤΩΝ ΓΙΑ ΤΗΝ ΣΧΕΔΙΑΣΗ ΤΩΝ ΚΥΜΑΤΟΜΟΡΦΩΝ 63
ΚΩΔΙΚΑΣ 32: WAVEFORMS_VIEWER.JS - Ο ΚΥΡΙΟΣ ΒΡΟΓΧΟΣ ΣΧΕΔΙΑΣΗΣ ΚΥΜΑΤΟΜΟΡΦΩΝ ΚΑΙ ΤΑ ΑΡΧΙΚΑ ΒΗΜΑΤΑ ΠΡΟΣΔΙΟΡΙΣΜΟΥ ΤΗΣ ΤΙΜΗΣ ΤΟΥ ΣΗΜΑΤΟΣ 64
ΚΩΔΙΚΑΣ 33: WAVEFORMS_VIEWER.JS - ΣΧΕΔΙΑΣΗ ΠΑΛΜΩΝ ΓΙΑ ΣΗΜΑΤΑ ΜΕ ΜΗΚΟΣ ΕΝΑ 65
ΚΩΔΙΚΑΣ 34: WAVEFORMS_VIEWER.JS - ΣΧΕΔΙΑΣΗ ΠΑΛΜΩΝ ΓΙΑ ΣΗΜΑΤΑ ΜΕ ΜΗΚΟΣ ΜΕΓΑΛΥΤΕΡΟ ΑΠΟ ΕΝΑ 66
ΚΩΔΙΚΑΣ 35: WAVEFORMS_VIEWER.JS - ΑΝΤΙΜΕΤΩΠΙΣΗ ΓΕΓΟΝΟΤΟΣ ΓΙΑ ΤΗΝ ΛΕΙΤΟΥΡΓΙΑ ΧΡΟΝΙΚΗΣ ΚΛΙΜΑΚΑΣ 67
ΚΩΔΙΚΑΣ 36: WAVEFORMS_VIEWER.JS - ΑΡΧΙΚΟΠΟΙΗΣΗ ΜΕΤΑΒΛΗΤΩΝ ΣΤΟ ΠΑΤΗΜΑ ΠΟΝΤΙΚΙΟΥ ΚΑΙ Η ΛΕΙΤΟΥΡΓΙΑ ΟΜΑΛΟΠΟΙΗΣΗΣ ΤΗΣ ΧΡΟΝΙΚΗΣ ΜΕΤΑΤΟΠΙΣΗΣ..... 68
ΚΩΔΙΚΑΣ 37: WAVEFORMS_VIEWER.JS - ΥΠΟΛΟΓΙΣΜΟΣ ΝΕΩΝ ΟΡΙΩΝ ΠΡΟΒΟΛΗΣ ΑΠΟΤΕΛΕΣΜΑΤΩΝ ΚΑΙ ΕΠΑΝΑΣΧΕΔΙΑΣΗ ΕΑΝ ΥΠΑΡΧΕΙ ΧΡΟΝΙΚΗ ΜΕΤΑΤΟΠΙΣΗ..... 69
ΚΩΔΙΚΑΣ 38: WAVEFORMS_VIEWER.JS - ΓΕΓΟΝΟΣ ΠΑΤΗΜΑΤΟΣ ΠΟΝΤΙΚΙΟΥ, ΑΡΧΙΚΟΠΟΙΗΣΗ ΚΑΙ ΥΠΟΛΟΓΙΣΜΟΣ ΜΕΤΑΒΛΗΤΩΝ ΓΙΑ ΤΟΝ ΜΟΝΙΜΟ ΣΗΜΑΤΟΔΟΤΗ ΕΠΙΠΛΕΩΝ ΠΛΗΡΟΦΟΡΙΩΝ 69
ΚΩΔΙΚΑΣ 39: WAVEFORMS_VIEWER.JS - ΕΥΡΕΣΗ ΤΙΜΗΣ ΓΙΑ ΚΑΘΕ ΣΗΜΑ ΓΙΑ ΤΟΝ ΜΟΝΙΜΟ ΣΗΜΑΤΟΔΟΤΗ ΕΠΙΠΛΕΩΝ ΠΛΗΡΟΦΟΡΙΩΝ 70
ΚΩΔΙΚΑΣ 40: WAVEFORMS_VIEWER.JS - ΕΜΦΑΝΙΣΗ ΤΙΜΗΣ ΣΗΜΑΤΟΣ ΣΤΟΝ ΜΟΝΙΜΟ ΣΗΜΑΤΟΔΟΤΗ ΕΠΙΠΛΕΟΝ ΠΛΗΡΟΦΟΡΙΩΝ 71
ΚΩΔΙΚΑΣ 41: WAVEFORMS_VIEWER.JS - ΔΗΜΙΟΥΡΓΙΑ ΚΑΙ ΜΕΤΑΦΟΡΤΩΣΗ ΕΙΚΟΝΑΣ ΑΠΟ ΤΟ ΣΤΟΙΧΕΙΟ 'CANVAS' .. 71
ΚΩΔΙΚΑΣ 42: ΕΝΤΟΛΕΣ ΕΓΚΑΤΑΣΤΑΣΗΣ ΤΟΥ ΠΡΟΓΡΑΜΜΑΤΟΣ GHDL..... 72
ΚΩΔΙΚΑΣ 43: ΕΝΤΟΛΕΣ ΕΓΚΑΤΑΣΤΑΣΗΣ ΤΗΣ MYSQL ΚΑΙ PHP 72
ΚΩΔΙΚΑΣ 44: ΕΝΤΟΛΗ ΕΓΚΑΤΑΣΤΑΣΗΣ, ΕΠΕΞΕΡΓΑΣΙΑΣ ΤΩΝ ΡΥΘΜΙΣΕΩΝ ΚΑΙ ΕΠΑΝΕΚΚΙΝΗΣΗΣ ΤΟΥ APACHE 72
ΚΩΔΙΚΑΣ 45: ΕΝΤΟΛΗ ΕΓΚΑΤΑΣΤΑΣΗΣ, ΕΠΕΞΕΡΓΑΣΙΑΣ ΤΩΝ ΡΥΘΜΙΣΕΩΝ ΚΑΙ ΕΚΚΙΝΗΣΗΣ ΤΟΥ NGINX..... 73
ΚΩΔΙΚΑΣ 46: ΕΝΤΟΛΕΣ ΡΥΘΜΙΣΗΣ ΤΗΣ ΒΑΣΗΣ ΔΕΔΟΜΕΝΩΝ 73

Κεφάλαιο 1 - Εισαγωγή

Στη σύγχρονη εποχή βλέπουμε τον ρόλο του διαδικτύου συνεχώς να αναπτύσσεται. Καθώς οι δυνατές ταχύτητες αγγίζουν συνεχώς νέα όρια, μας επιτρέπεται η χρήση του σε νέες εφαρμογές. Παράλληλα η ανάπτυξη εξυπηρετητών με αναπτυγμένες προδιαγραφές, που αρκετές φορές ξεπερνούν σε εντυπωσιακό επίπεδο αυτές των προσωπικών υπολογιστών, επιτρέπουν την λειτουργία απαιτητικών εφαρμογών σε συσκευές με περιορισμένους πόρους. Το γεγονός αυτό ενισχύεται ακόμη περισσότερο όταν γίνεται η ταυτόχρονη χρήση πολλαπλών εξυπηρετητών για την παράλληλη εκτέλεση μιας εφαρμογής ή αποθήκευση δεδομένων. Επίσης κατά την πορεία της προόδου όλο και περισσότεροι χρήστες χρησιμοποιούν κινητές συσκευές για να αποκτήσουν πρόσβαση σε εφαρμογές και προγράμματα που τους προσφέρουν ένα περιβάλλον αλληλεπίδρασης ενώ οι απαιτητικές λειτουργίες τους τρέχουν στον εξυπηρετητή.

Έχοντας υπόψη τα παραπάνω, γίνεται πλέον κατανοητός ο λόγος που αναπτύξαμε την εφαρμογή μας σε ένα διαδικτυακό περιβάλλον. Η επιλογή αυτή μας επέτρεψε να θέσουμε στόχους τόσο προς την λειτουργικότητα όσο και προς την προσιτότητα της εφαρμογής. Φυσικά η προσιτότητα της εφαρμογής επέφερε και την αναπόφευκτη επαφή της με κακόβουλες αλληλεπιδράσεις κι έτσι δημιουργήθηκε η ανάγκη για λειτουργίες και ελέγχους ασφαλείας. Όπως γίνεται πλέον φανερό, οι ανεπτυγμένες δυνατότητες που προσφέρει το διαδικτυακό περιβάλλον, την εκθέτουν σε επιπλέον κινδύνους που πρέπει να καλυφθούν. Σε επόμενα κεφάλαια γίνεται λεπτομερής αναφορά στον τρόπο υλοποίησης της εφαρμογής όπου γίνονται κατανοητές και οι επιλογές που κάναμε για τον τρόπο αντιμετώπισης των παραπάνω απαιτήσεων. Έχοντας υπόψη τόσο την ύπαρξη αυτών των απαιτήσεων όσο και των δυνατοτήτων που μας προσφέρονται, γίνεται φανερός ο λόγος που μια διαδικτυακή εφαρμογή έχει έναν ριζικά διαφορετικό τρόπο ανάπτυξης από μία τοπική εφαρμογή ακόμη και όταν εκτελούν παρόμοιες λειτουργίες.

Η πορεία εξέλιξης των διαδικτυακών εφαρμογών βρίσκεται ακόμη σε αρχικά στάδια. Μπορούμε να περιμένουμε, στο άμεσο μέλλον, σημαντικές αλλαγές και βελτιώσεις που θα

αγγίξουν όλες τις εφαρμογές σε αυτό το περιβάλλον. Στο τέλος της εργασίας γίνονται αναφορές για πιθανές μελλοντικές επεκτάσεις. Αυτές οι επεκτάσεις αφορούν κυρίως τις ίδιες τις λειτουργίες του εργαλείου. Νέες όμως εξελίξεις τόσο στις δυνατότητες εξυπηρετητών όσο και στις ταχύτητες του δικτύου μπορούν να ανοίξουν τον δρόμο για λειτουργίες που αυτή τη στιγμή θα απαιτούσαν υπερβολικούς πόρους. Παράλληλα, εξελίξεις στις χρησιμοποιούμενες τεχνολογίες δημιουργούν ήδη νέες δυνατότητες όπως η χρήση HTML 5 και του στοιχείου 'canvas' που μας επιτρέπουν την χρήση της κάρτας γραφικών σε εφαρμογές περιηγητών απελευθερώνοντας τον επεξεργαστή του χρήστη που δεν είναι εξειδικευμένος για τέτοιες λειτουργίες.

Στην εργασία αυτή αυτό θα αναλύσουμε την εφαρμογή που αναπτύξαμε για την επεξεργασία, μεταγλώττιση και προσομοίωση HDL προγραμμάτων και το βοηθητικό εργαλείο της για την σχεδίαση κυματομορφών σημάτων που προκύπτουν από την προσομοίωση καθώς και όλα τα στάδια υλοποίησής τους από την ιδέα ως τη δημιουργία. Θα δούμε αναλυτικά τους λόγους που σχεδιάστηκε αυτή η εφαρμογή, τα εργαλεία που χρησιμοποιήθηκαν, τον τρόπο υλοποίησης καθώς και τον τρόπο αντιμετώπισης απαιτήσεων που προέκυψαν και τέλος τον τρόπο λειτουργίας της εφαρμογής. Συνολικά περιγράφεται η ολοκληρωμένη διαδικασία ανάπτυξής και σχεδίασης μίας διαδικτυακής εφαρμογής.

1.1 Κίνητρα και Στόχοι Υλοποίησης

Ο προγραμματισμός σε HDL έχει ως στόχο την δημιουργία προγραμμάτων για την περιγραφή κυκλωμάτων. Ένα πρόγραμμα HDL περιγράφει ένα ψηφιακό σύστημα και μας επιτρέπει τόσο την ανάλυση και προσομοίωσή του πριν το εισάγουμε στο υλικό όσο και την σύνθεσή του σε προγραμματιζόμενες πλακέτες χωρίς την απαίτηση δημιουργίας νέου ολοκληρωμένου κυκλώματος. Όπως γίνεται προφανές, ένα πολύ σημαντικό ορόσημο αυτής της διαδικασίας είναι η προσομοίωση του κώδικα. Μέσω της προσομοίωσης επιβεβαιώνεται η σωστή λειτουργία του προγράμματος άμεσα και αποτελεσματικά σε ένα εικονικό περιβάλλον. Υπάρχουν διάφορες γλώσσες HDL όπως η VHDL (Very High Speed Integrated Circuit HDL, πολύ υψηλής ταχύτητας ολοκληρωμένων κυκλωμάτων HDL) και η Verilog, εμείς θα ασχοληθούμε με την VHDL. Συγκεκριμένα για την γλώσσα VHDL τα διαθέσιμα προγράμματα που επιτρέπουν την παραπάνω διαδικασία είναι κυρίως εμπορικά και πάντα τοπικά. Όπως γίνεται εύκολα αντιληπτό αυτές οι ιδιότητες προκαλούν πολλούς περιορισμούς σε μια τόσο σημαντική εφαρμογή. Έτσι προέκυψε η ανάγκη για την δημιουργία μιας δωρεάν και ανοιχτού κώδικα εφαρμογής που θα προσφέρει τα απαραίτητα εργαλεία σε ένα διαδικτυακό περιβάλλον.

Η εφαρμογή που παρουσιάζεται έχει ως πρώτο στόχο την διευκόλυνση τη δημιουργία και τον έλεγχο προγραμμάτων HDL. Πρόκειται για μια ιστοσελίδα που επιτρέπει την διατήρηση προγραμμάτων στον εξυπηρετητή καθώς και την μεταγλώττιση και προσομοίωσή τους. Υποστηρίζονται τρεις οντότητες χρηστών με διαφορετικά δικαιώματα, τον επισκέπτη, εγγεγραμμένου χρήστη και διαχειριστή. Επιτρέπει την δημιουργία και επεξεργασία αρχείων σε κειμενογράφο με ειδικές επισημάνσεις, την ανάλυση των αποτελεσμάτων σε ενσωματωμένο εργαλείο προβολής κυματομορφών καθώς και την διαμοίραση αρχείων ως εξαρτήματα μεταξύ χρηστών. Γενικά, η εφαρμογή μας στοχεύει να προσφέρει σε προγραμματιστές HDL την δυνατότητα ανάπτυξης και ανάλυσης των προγραμμάτων τους σε ένα διαδικτυακό περιβάλλον.

Η εμπορικότητα ενός προγράμματος είναι φυσικά μια αναμενόμενη κατάσταση όταν αυτό αναπτύσσεται από εταιρίες που έχουν ως σκοπό το κέρδος. Τέτοια προγράμματα στοχεύουν σε κοινό παρόμοιας εμβέλειας, συνήθως άλλες εταιρίες που τα χρησιμοποιούν για την παραγωγή δικών τους προϊόντων ή υπηρεσιών. Έτσι δημιουργείται όμως ένα κενό προς το υπόλοιπο ενδιαφερόμενο κοινό όπως η ακαδημαϊκή κοινότητα όπου σκοπός είναι η έρευνα, η ανάπτυξη και η εκπαίδευση και όχι το κέρδος. Σε ένα τέτοιο κοινό τα ιδανικά προγράμματα δεν είναι μόνο δωρεάν αλλά και ανοιχτού κώδικα ώστε να μπορούν αυτά να μελετηθούν και να αναπτυχθούν περαιτέρω. Έτσι γίνεται πλέον φανερός ο πρώτος στόχος και το βασικό κίνητρο για την ανάπτυξη της εφαρμογής μας.

Ο δεύτερος στόχος της εφαρμογής είναι η διαδικτυακή της φύση. Η πρώτη εμφανής επίδραση αυτής της φύσης είναι η προσιτότητα. Η εφαρμογή πλέον γίνεται προσβάσιμη από οποιονδήποτε υπολογιστή με σύνδεση στο διαδίκτυο και υποστήριξη ενός σύγχρονου περιηγητή χωρίς την απαίτηση εγκατάστασης επιπλέον προγραμμάτων και βοηθητικών βιβλιοθηκών. Επίσης οι απαιτήσεις πόρων για την αλληλεπίδραση με την εφαρμογή είναι σημαντικά λιγότερες από ένα τοπικό πρόγραμμα. Αυτό προσθέτει πολλές φορητές συσκευές στους τρόπους πρόσβασης της εφαρμογής. Κάτι τέτοιο θα προκαλούσε σημαντικές δυσκολίες σε ένα τοπικό πρόγραμμα καθώς θα απαιτούνταν η υποστήριξη των διαφορετικών λειτουργικών συστημάτων. Έτσι γίνεται εμφανές μια ακόμη υπεροχή του διαδικτυακού περιβάλλοντος. Καθώς όμως αυτό το περιβάλλον μας επιτρέπει από την φύση του την αλληλεπίδραση των χρηστών του, επιτεύχθηκαν στόχοι που δεν προβλέφθηκαν στην αρχική σχεδίαση. Η διαμοίραση συχνά χρησιμοποιούμενων εξαρτημάτων, όπως ένας αθροιστής, είναι πλέον μια απλή διαδικασία για τον κάθε χρήστη. Παρόμοια, η συλλογική ανάπτυξη προγραμμάτων γίνεται εύκολα προσθέτοντας συγγραφείς σε κάποιο έργο. Αυτές οι διαδικασίες μας επέτρεψαν να θέσουμε στόχους που μια τοπική εφαρμογή θα μπορούσε να φτάσει μόνο με σημαντικές προσθήκες ή εξωτερικές λειτουργίες, όπως έναν διαδικτυακό τόπο συζήτησης.

Ξεκινήσαμε την ανάπτυξη της εφαρμογής για να καλύψουμε ένα κενό. Το κενό αυτό βρισκόταν τόσο στον τρόπο διάθεσης παρόμοιων εφαρμογών όσο και στον τρόπο που καλύπτουν τις απαιτούμενες λειτουργίες τους. Οι στόχοι που τέθηκαν κατά την σχεδίαση δεν ήταν όμως οι μόνοι που καλύφθηκαν. Κατά την υλοποίηση της εφαρμογής αντιληφθήκαμε τη δυνατότητα υιοθέτησης νέων στόχων που προέκυψαν από την φύση της. Η τελική μορφή της εφαρμογής καλύπτει πλέον τις ανάγκες του προγραμματιστή αλλά προσφέρει ταυτόχρονα και νέες δυνατότητες για την διαδικασία προσομοίωσης και δοκιμής του προγράμματός του.

1.2 Σχετικά Εργαλεία

Η εφαρμογή που παρουσιάζεται, όπως εξηγήθηκε παραπάνω, δημιουργήθηκε για να καλύψει ένα κενό. Έτσι γίνεται κατανοητό, ότι δεν υπάρχουν άλλα εργαλεία που επιτελούν το ίδιο έργο με τον ίδιο τρόπο. Παρόλα αυτά υπάρχουν εργαλεία που έχουν τον ίδιο στόχο σε διαφορετικό περιβάλλον καθώς και εργαλεία στο ίδιο περιβάλλον που όμως δεν καλύπτουν όλους τους στόχους που έχουμε θέσει εμείς. Τέτοια εργαλεία έχουν πλέον θέσει κάποια πρότυπα που είναι οικεία προς τους χρήστες και αποτελέσαν ταυτόχρονα έμπνευση αλλά και μέτρα σύγκρισης για την εφαρμογή μας.

Ένα από τα πιο διαδεδομένα και συχνά χρησιμοποιημένα προγράμματα είναι το ModelSim [1] από την εταιρία Mentor Graphics. Το ModelSim είναι ένα τοπικό πρόγραμμα που επιτρέπει την προσομοίωση και ανάλυση αποτελεσμάτων για τις γλώσσες VHDL, Verilog και SystemC [2]. Οι σύγχρονες εκδόσεις που προσφέρονται υποστηρίζουν μόνο Microsoft Windows και ενώ δίνεται δοκιμαστική έκδοση με περιορισμό χρήσης στις 21 μέρες, η ολοκληρωμένης έκδοσης απαιτεί την αγορά άδειας χρήσης. Παρόλα αυτά το ModelSim προσφέρει σε έναν εξοικειωμένο χρήστη την δυνατότητα ανάλυσης του παραγόμενου κυκλώματος σε βάθος, ενώ υπάρχει πλούτος διδακτικού υλικού για τους αρχάριους χρήστες. Αξιοσημείωτο επίσης είναι το πρόγραμμα Vivado Design Suite [3] από την εταιρία Xilinx. Το συγκεκριμένο πρόγραμμα βασίζεται στο ISE Simulator [4], επίσης από την Xilinx, και σκοπεύει να το αντικαταστήσει. Το Vivado Design Suite δεν περιορίζεται στην προσομοίωση αλλά περιλαμβάνει την ολοκληρωμένη διαδικασία από την συγγραφή ως στην σύνθεση προγραμμάτων σε πλακέτα. Υποστηρίζει VHDL, Verilog και SystemC ενώ μπορεί να λειτουργήσει σε Microsoft Windows αλλά και Linux με περιορισμό όμως σε συστήματα των 64 bit. Το Vivado Design Suite προσφέρεται σε πολλές διαφορετικές εκδόσεις με διαφορετικές δυνατότητες που αντικατοπτρίζουν και το κόστος της κάθε έκδοσης αλλά και δοκιμαστική έκδοση με όλες τις δυνατότητες η οποία υπόκειται σε περιορισμό των 30 ημερών. Υποστηρίζεται ότι το Vivado Design Suite έχοντας ενσωματωμένο προσομοιωτή μπορεί να προσφέρει προσομοίωση μέχρι και τρεις φορές γρηγορότερα από το ISE Simulator [5]. Αξίζει να σημειωθεί ότι το ISE Simulator χρησιμοποιεί το ModelSim για προσομοίωση. Κλείνοντας τις αναφορές σε τοπικά προγράμματα, θα αρκεστούμε σε μια επιφανειακή ανάλυση του Icarus Verilog [6]. Το συγκεκριμένο πρόγραμμα αποτελείται από ένα εργαλείο προσομοίωσης και σύνθεσης με περιορισμένες δυνατότητες. Υποστηρίζει μόνο την γλώσσα Verilog αλλά μπορεί να χρησιμοποιηθεί σε Microsoft Windows, Linux, OpenSolaris, AIX, FreeBSD και Mac OS X. Το Icarus Verilog στοχεύει κυρίως στην ίδια την προσομοίωση κώδικα και προτρέπει στην χρήση εξωτερικών προγραμμάτων διαδικασίες όπως η προεπισκόπηση κυματομορφών με GTKWave [7] η ακόμη και η ολοκληρωμένη σύνθεση προγραμμάτων με Yosys [8]. Γίνεται πλέον φανερό ότι υπάρχει πλήθος τοπικών προγραμμάτων που αποσκοπούν στην μεταγλώττιση και προσομοίωση HDL προγραμμάτων με μεγάλη ποικιλία δυνατοτήτων που προσφέρονται .

Στο περιβάλλον του διαδικτύου υπάρχουν επίσης πολλά εργαλεία που αποσκοπούν σε παρόμοιες λειτουργίες. Όπως είδαμε στην προηγούμενη παράγραφο, το Icarus Verilog προσφέρει περιορισμένες δυνατότητες σε τοπικό περιβάλλον. Ταυτόχρονα όμως επιτρέπει λόγο τις φύσης του ως προσομοιωτής σε τερματικό, την χρήση του στο εργαλείο iVerilog [9]. Το εργαλείο αυτό αξιοποιεί την χρήση του προσομοιωτή Icarus Verilog προσφέροντας την δυνατότητα στον χρήστη να δοκιμάσει και να εκτελέσει τον κώδικά του με την προσιτότητα τόσο στην τοποθεσία όσο και στις συσκευές που προσφέρει το διαδίκτυο. Παράλληλα όμως, το iVerilog μοιράζεται και τους σημαντικότερους περιορισμούς του Icarus Verilog. Συγκεκριμένα, παρατηρείται αρχικά ότι δεν υποστηρίζεται κάποιο είδος προβολής κυματομορφών και ο μόνος τρόπος προβολής αποτελεσμάτων είναι η έξοδός τους προς το τερματικό. Επίσης δεν υπάρχει υποστήριξη για την αποθήκευση προγραμμάτων ή την χρήση διαφορετικών αρχείων. Ενώ φυσικά, υποστηρίζεται και πάλι μόνο η γλώσσα Verilog. Ένα πιο ολοκληρωμένο εργαλείο είναι το EDA Playground [10]. Το εργαλείο αυτό υποστηρίζει αρκετές διαφορετικές γλώσσες τόσο στην περιγραφή υλικού όσο και σε άλλους τομείς. Ταυτόχρονα προσφέρει την επιλογή μεταξύ διαφόρων εργαλείων προσομοίωσης, προσθήκη βιβλιοθηκών, επιλογές μεταγλώττισης αλλά και αποτελεσματικό εργαλείο προβολής κυματομορφών.

Παράλληλα δίνει την δυνατότητα σε συνδεμένους χρήστες να διατηρήσουν το έργο τους αλλά προσφέρει ακόμα και μια δοκιμαστική εμπειρία διαμοιρασμού και παράλληλης προβολής κώδικα. Το EDA Playground είναι αντικειμενικά το κοντινότερο εργαλείο σε αυτό που αναπτύξαμε αλλά και πάλι βλέπουμε σημαντικές διαφορές. Το EDA Playground δεν υποστηρίζει δωρεάν προσομοιωτές για VHDL και παρότι η χρήση του είναι δωρεάν, δεν είναι ανοιχτού κώδικα. Η λειτουργία του για συνεργασία προγραμματιστών βασίζεται στην λογική της επίδειξης κώδικα και όχι στην παράλληλη συγγραφή του. Παράλληλα βλέπουμε ότι ενώ υπάρχουν συγκεκριμένα παραδείγματα και εξωτερικές βιβλιοθήκες, δεν υπάρχει τρόπος να διαμοιραστεί κάποιος συχνά επαναλαμβανόμενος κώδικας μεταξύ χρηστών. Ενώ το εργαλείο παρουσίασης κυματομορφών είναι σίγουρα αποτελεσματικό, βλέπουμε αρκετές λεπτομέρειες που το διαχωρίζουν από αυτό που αναπτύξαμε. Συγκεκριμένα, δεν υπάρχει τρόπος ανάπτυξης σημάτων πάνω από ένα bit, δεν υπάρχει συνεχής δείκτης αναφοράς τιμών ώστε να υποβοηθήσει την ανάλυση του σήματος, ο δείκτης που θέτει ο χρήστης βρίσκεται μόνο στην αλλαγή τιμής του συγκεκριμένου σήματος με αποτέλεσμα να μην συγκρίνει τα υπόλοιπα σήματα στην ίδια χρονική στιγμή. Γίνεται προφανές ότι οι λεπτομέρειες αυτές είναι αρκετά τεχνικές και χρειάζονται ίσως ιδιαίτερη προσοχή από κάποιο σημείο και μετά για να γίνουν εμφανείς. Αυτές όμως οι λεπτομέρειες τελικά δημιουργούν δύο διαφορετικά συστήματα που προσφέρουν τα ίδια βασικά χαρακτηριστικά αλλά με διαφορετική προσέγγιση και καταλήγουν να αντικατοπτρίζουν διαφορετικό κοινό χρηστών.

Οι εφαρμογές που παρουσιάστηκαν παραπάνω αποτελούν τελικά το περιβάλλον στο οποίο αναπτύχθηκε η δική μας. Παίρνοντας από τις πιο ενδιαφέρουσες ιδέες σημαντικά μαθήματα, καταφέραμε να αποστάξουμε τα καλύτερα στοιχεία τους και να τα προσαρμόσουμε στους δικούς μας στόχους. Ξεπεράσαμε δυσκολίες στις οποίες άλλα εργαλεία έχουν σταματήσει την ανάπτυξη τους και ιδρύσαμε νέους τομείς στον τρόπο που ακολουθείται η διαδικασία προσομοίωσης και ανάλυσης HDL προγραμμάτων. Συγκεκριμένα, ξεπεράσαμε το φράγμα εμφάνισης αποτελεσμάτων σε μορφές κυματομορφών δημιουργώντας από την αρχή το δικό μας εργαλείο που ανταποκρίνεται αποτελεσματικά σε ένα διαδικτυακό περιβάλλον. Παράλληλα, δώσαμε στους χρήστες νέους τρόπους αλληλεπίδρασης και θέσαμε νέα όρια στον συλλογικό προγραμματισμό. Όλα αυτά, κρατώντας ταυτόχρονα τις βασικές λειτουργίες που απαιτούνται, όπως η χρήση κειμενογράφου, διατήρηση αρχείων και έργων καθώς και η ενημέρωση σφαλμάτων κατά την μεταγλώττιση. Δημιουργήσαμε τελικά μια ολοκληρωμένη εφαρμογή, που προκαλεί τα εργαλεία του είδους του και επιτρέπει τους προγραμματιστές HDL να θέσουν ανώτερες απαιτήσεις από αυτά.

1.3 Διάρθρωση Βιβλίου

Το υπόλοιπο βιβλίο διαρθρώνεται ως εξής. Στο κεφάλαιο 2 γίνεται αναφορά στο θεωρητικό υπόβαθρο που απαιτείται για την κατανόηση της υλοποίησης του εργαλείου καθώς και στα βασικά εργαλεία που χρησιμοποιήθηκαν. Στο κεφάλαιο 3 γίνεται η λεπτομερής ανάλυση της σχεδίασης και υλοποίησης της εφαρμογής. Αρχικά εξηγείται η διάρθρωση της βάσης δεδομένων και η ύπαρξη κάθε πίνακα, έπειτα η διάρθρωση των απαιτούμενων καταλόγων. Το κεφάλαιο συνεχίζει με την ανάπτυξη και υλοποίηση της εφαρμογής όπου γίνεται κατανοητός ο τρόπος που υλοποιήθηκαν πολλές από τις βασικές λειτουργίες, έπειτα εξηγείται η υλοποίηση του εργαλείου σχεδίασης κυματομορφών και τέλος δίνονται οι οδηγίες εγκατάστασης. Στο

κεφάλαιο 4 επεξηγούνται οι λειτουργίες που προσφέρει η εφαρμογή. Αρχικά γίνεται μια ανασκόπηση στις λειτουργίες σχετικές με τα έργα ή προγράμματα, συνεχίζοντας στις λειτουργίες σχετικά με τα εξαρτήματα ή βιβλιοθήκες και τέλος, αναφέρονται τα διαφορετικά επίπεδα χρηστών και οι διαφορετικές τους δυνατότητες. Το κεφάλαιο 5 είναι ο επίλογος του βιβλίου όπου παρατίθενται τα αποτελέσματα και προσφέρονται μερικές πιθανές επεκτάσεις της εφαρμογής στο μέλλον.

Κεφάλαιο 2 - Θεωρητικό Υπόβαθρο

Όπως έγινε φανερό στο προηγούμενο κεφάλαιο, οι στόχοι που θέσαμε για την εφαρμογή μας, έκαναν το διαδικτυακό περιβάλλον μια άμεση επιλογή. Η ανάπτυξη μιας εφαρμογής σε αυτό το περιβάλλον μπορεί να ακολουθήσει πολλές διαδρομές. Στοχεύοντας σε μια εφαρμογή με δωρεάν διάθεση και ανοιχτού κώδικα, επιλέχθηκε η διαδρομή μέσα από εργαλεία παρόμοιας φύσης.

Σε αυτό το κεφάλαιο λοιπόν, θα κάνουμε μια ανασκόπηση των εργαλείων που χρησιμοποιήθηκαν για την ανάπτυξη της εφαρμογής μας. Αρχικά εξετάζονται τα βασικά εργαλεία διαδικτυακού προγραμματισμού στα οποία βασίσαμε την ανάπτυξη της εφαρμογής και έπειτα περιγράφονται τα εξωτερικά εργαλεία που ενσωματώσαμε για την αποτελεσματική λειτουργία της.

2.1 Βασικά Εργαλεία Προγραμματισμού Διαδικτύου

Τα βασικά εργαλεία που παρουσιάζονται παρακάτω είναι οι γλώσσες προγραμματισμού με τις οποίες αναπτύξαμε την εφαρμογή μας. Αυτές αποτελούν τις βασικές γλώσσες διαδικτυακού προγραμματισμού που χρησιμοποιούνται κατά κόρον για την ανάπτυξη ιστοσελίδων απλών ή περίπλοκων λειτουργιών.

2.1.1 HTML και CSS

Η HTML (HyperText Markup Language, Γλώσσα Μορφοποίησης Υπερκειμένου) [11] χρησιμοποιείται κατά κόρων για την δημιουργία ιστοσελίδων. Βασίζεται στην χρήση στοιχείων που θέτονται από ετικέτες. Τα στοιχεία αυτά ποικίλουν από ορισμός του τίτλου της σελίδας, ορισμός παραγράφων και νέων γραμμών ως την σήμανση μιας εικόνας και στοιχείων σχεδιασμού γραφικών. Με την χρήση αυτών των στοιχείων, δίνεται η δυνατότητα περιγραφής μια σελίδας με τρόπο κατανοητό από έναν περιηγητή. Έτσι η HTML χρησιμοποιείται ως το βασικό δομικό στοιχείο μιας ιστοσελίδας.

Η έκδοση που χρησιμοποιήσαμε στην εφαρμογή μας είναι η HTML5 και αποτελεί την σύγχρονη έκδοση ενώ περιέχει απαραίτητα στοιχεία για τις λειτουργίες που απαιτούμε. Η έκδοση αυτή προσφέρει νέα στοιχεία και γνωρίσματα στοιχείων αλλά ταυτόχρονα βελτιώνει υπάρχοντα στοιχεία. Οι διαφορές από προηγούμενες εκδόσεις είναι πολλές [12] και ξεφεύγουν από το πλαίσιο αυτού του βιβλίου. Παρόλα αυτά, μία πολύ σημαντική προσθήκη που αξίζει να αναφερθεί είναι αυτή του στοιχείου ‘canvas’. Με το συγκεκριμένο στοιχείο μας επιτρέπεται η χρήση γραφικών απευθείας στον περιηγητή χωρίς την χρήση εξωτερικών προγραμμάτων. Η αξιοποίηση αυτού του στοιχείου αποτέλεσε την βάση δημιουργίας του εργαλείου προβολής κυματομορφών

Η HTML λοιπών, μας επιτρέπει την δημιουργία των βασικών στοιχείων που χρησιμοποιεί μια ιστοσελίδα. Τα στοιχεία αυτά όμως πρέπει να παρουσιαστούν με συγκεκριμένο τρόπο ώστε να δημιουργηθεί ένα φιλικό προς τον χρήστη περιβάλλον. Την παρουσίαση των στοιχείων αναλαμβάνει μια άλλα γλώσσα, η CSS (Cascading Style Sheets, Διαδοχικά Φύλλα Στυλ).

Η γλώσσα CSS [13] χρησιμοποιείται για την περιγραφή του τρόπου εμφάνισης των HTML στοιχείων. Προσφέρει στον περιηγητή πληροφορίες όπως το χρώμα, τις αποστάσεις, το μέγεθος ή ακόμη και την ορατότητα του κάθε στοιχείου. Η γλώσσα CSS μας επιτρέπει την δημιουργία διαφορετικών κλάσεων με ξεχωριστούς κανόνες. Στην συνέχεια απονέμουμε αυτές τις κλάσεις στα διάφορα στοιχεία HTML προσδιορίζοντας τον τρόπο εμφάνισής τους. Σε σπάνιες περιπτώσεις χρησιμοποιούμε κανόνες απευθείας σε στοιχεία. Αυτό γίνεται συνήθως σε στοιχεία που εμφανίζονται μία φορά στην ιστοσελίδα όπως το στοιχείο ‘body’. Σε άλλες περιπτώσεις έχουμε κανόνες εξαίρεσης σε διάφορα στοιχεία. Τα στοιχεία αυτά συνήθως αναγνωρίζονται με μοναδική επισήμανση με το γνώρισμα ‘id’. Η χρήση κλάσεων για την εφαρμογή κανόνων μας επιτρέπει τόσο την εύκολη συντήρηση κώδικα, όσο και την ελαχιστοποίηση της πληροφορίας που πρέπει να αποσταλθεί στον χρήστη.

Συγκεκριμένα στην εφαρμογή που αναπτύξαμε, χρησιμοποιούμε την έκδοση CSS3 η οποία μας δίνει την δυνατότητα επέκτασης των λειτουργιών της με την χρήση ενοτήτων. Με αυτές τις ενότητες μπορούμε να προσθέσουμε νέους κανόνες στις κλάσεις που χρησιμοποιούμε. Εκτός όμως από τους νέους κανόνες, η γλώσσα CSS3 μας δίνει την δυνατότητα για νέες ψευδοκλάσεις. Οι ψευδοκλάσεις αυτές μας επιτρέπουν την επέκταση κλάσεων για την επιλογή συγκεκριμένων στοιχείων που ανήκουν σε αυτές. Ένα παράδειγμα που χρησιμοποιείται ευρέος είναι η ψευδοκλάση ‘hover’ όπου τα στοιχεία που ανήκουν σε αυτή είναι τα στοιχεία στα οποία βρίσκεται ο δείκτης ποντικιού. Κάποιες από αυτές τις ψευδοκλάσεις εμφανίστηκαν σε παλαιότερες εκδόσεις αλλά με την CSS3 έγινε

επαναπροσδιορισμός αυτών που προϋπήρχαν και είσοδος νέων. Έτσι η γλώσσα CSS3 μας δίνει πλέον νέες δυνατότητες ενώ παράλληλα εισάγονται συνεχώς νέες ενότητες που οδηγούν σε περαιτέρω ανάπτυξη των λειτουργιών της.

Οι γλώσσες HTML και CSS μας επιτρέπουν την εισαγωγή στοιχείων και την προσαρμογή στον τρόπο εμφάνισής τους. Οι δυνατότητες αυτές είναι τα βασικά δομικά στοιχεία κάθε ιστοσελίδας. Μια ιστοσελίδα μπορεί να αποτελείται μόνο από αυτά τα στοιχεία εάν ο σκοπός της είναι μια στατική λειτουργία. Μια εφαρμογή διαδικτύου όμως απαιτεί την αλληλεπίδραση με τον χρήστη και μια δυναμική διεπαφή.

2.1.2 JavaScript

Η γλώσσα JavaScript [14] είναι μια διερμηνευμένη γλώσσα προγραμματισμού που επιτρέπει, στα πλαίσια του προγραμματισμού διαδικτύου, την εκτέλεση προγραμμάτων μετά την μεταφόρτωση της σελίδα από τον χρήστη. Η ιδιότητα αυτή μας δίνει την δυνατότητα να προσθέσουμε μία δυναμική και διαδραστική αλληλεπίδραση του χρήστη με την ιστοσελίδα.

Στην εφαρμογή που αναπτύξαμε, ο τρόπος που χρησιμοποιούμε την γλώσσα JavaScript είναι κυρίως χειρισμού γεγονότων. Κατά την περιήγηση στην σελίδα ο χρήστης δημιουργεί διάφορα γεγονότα. Ένα γεγονός μπορεί να είναι η αποστολή μίας φόρμας ή το πέρασμα του δείκτη ποντικιού πάνω από ένα στοιχείο HTML. Στην εφαρμογή μας χρησιμοποιούμε τόσο το αρχικό όσο και το πρότυπο μοντέλο γεγονότων. Όπως περιγράφει ο συγγραφέας στο [15], το αρχικό μοντέλο γεγονότων μας προσφέρει αμεσότητα, ταχύτητα και υποστηρίζεται από του περισσότερους περιηγητές ενώ το πρότυπο μοντέλο γεγονότων μας προσφέρει μεγαλύτερες δυνατότητες και υποστηρίζεται από τους περισσότερους σύγχρονους περιηγητές. Υπάρχουν δύο βασικοί τρόποι που αντιμετωπίζονται τα γεγονότα. Ο πρώτος τρόπος είναι η εκτέλεση μιας λειτουργίας όπως η εμφάνιση ενός στοιχείου HTML, η επιβεβαίωση των εισαγόμενων στοιχείων σε μία φόρμα ή η σχεδίαση της κυματομορφής του επιλεγμένου κύματος. Με αυτόν τον τρόπο επιτρέπουμε στον χρήστη την αξιοποίηση λειτουργιών και επιβεβαιώνουμε την σωστή λειτουργία τους. Ο δεύτερος τρόπος που αντιμετωπίζονται τα γεγονότα είναι με την χρήση της τεχνικής AJAX (Asynchronous JavaScript and Extensible Markup Language, Ασύγχρονη JavaScript και Επεκτάσιμη Γλώσσα Σήμανσης). Η τεχνική αυτή πρόκειται για την ασύγχρονη ανταλλαγή δεδομένων μεταξύ ενός χρήστη και ενός εξυπηρετητή. Χρησιμοποιώντας αυτόν τον τρόπο, μπορούμε να αντιμετωπίσουμε γεγονότα όπως η αποθήκευση αλλαγών σε ένα αρχείο χωρίς την ανάγκη για την αποστολή μιας φόρμας που θα προκαλούσε επανάληψη φόρτωσης της σελίδας με αποτέλεσμα την διακοπή συγγραφής. Με παρόμοιο τρόπο, η ίδια τεχνική μας επιτρέπει την ανάκτηση στοιχείων από τον εξυπηρετητή επιτρέποντάς μας για παράδειγμα την άμεση επίδειξη αποτελεσμάτων σε μία φόρμα αναζήτησης. Γίνεται εύκολα φανερό ότι αυτός ο τρόπος αντιμετώπισης γεγονότων μας δίνει την δυνατότητα να προσφέρουμε στον χρήστη μια πιο άμεση αλληλεπίδραση χωρίς περιττές διακοπές.

Με την παρουσίαση της γλώσσας JavaScript ολοκληρώνουμε την παρουσίαση των βασικών εργαλείων που χρησιμοποιούμε στην πλευρά του χρήστη. Οι γλώσσες HTML και CSS παρουσιάζουν τα στοιχεία αλληλεπίδρασης και η γλώσσα JavaScript προσθέτει δυναμικές λειτουργίες σε αυτά. Έτσι δημιουργούμε την συνολική διεπαφή που προσφέρεται στον χρήστη.

Παράλληλα όμως, μία ολοκληρωμένη εφαρμογή, απαιτεί και έναν λειτουργικό εξυπηρετητή. Παρακάτω παρουσιάζονται οι γλώσσες προγραμματισμού που αναλαμβάνουν αυτό το καθήκον και αποτελούν τα βασικά συστατικά της εφαρμογής μας στο περιβάλλον του εξυπηρετητή.

2.1.3 PHP

Η γλώσσα PHP (PHP: Hypertext Preprocessor, PHP: Προεπεξεργαστής Υπερκειμένου) [16] είναι μία γλώσσα σεναρίου (script) που χρησιμοποιείται ευρέως στον προγραμματισμό διαδικτύου για να προσφέρει λειτουργίες στον εξυπηρετητή. Η γλώσσα αρχικά αναπτύχθηκε από τον Rasmus Lerdorf με σκοπό την συλλογή στατιστικών δεδομένων για την επισκεψιμότητα της προσωπικής του ιστοσελίδας. Κατά την $2^{η}$ έκδοση έγινε η προσθήκη ενός εργαλείου για την αλληλεπίδραση με βάση δεδομένων ενώ στην $3^{η}$ πλέον έκδοση είχαμε την συσσώρευση των λειτουργιών και επαναπροσδιορισμό τους σε μια γλώσσα προγραμματισμού [17]. Στην εφαρμογή που αναπτύξαμε χρησιμοποιούμε την γλώσσα PHP 5 και συγκεκριμένα την έκδοση 5.5.33.

Όπως αναφέραμε παραπάνω, η χρήση της γλώσσας PHP στοχεύει στην προσθήκη λειτουργιών στον εξυπηρετητή. Προσφέρει τις βασικές λειτουργίες μίας γλώσσας προγραμματισμού με επιπλέον εξειδίκευση στον προγραμματισμό διαδικτύου. Δίνεται έμφαση στην άμεση και ασφαλή διάδραση με μία βάση δεδομένων ενώ ταυτόχρονα προσφέρονται ενσωματωμένες λειτουργίες και προκαθορισμένες μεταβλητές για διαδικτυακά δεδομένα, όπως η διεύθυνση του εξυπηρετητή ή του επισκέπτη. Όπως φαίνεται στην Εικόνα 1, η PHP μπορεί να βρεθεί ένθετη σε κώδικα HTML αρκεί ο κώδικάς της να αρχικοποιείται με την σήμανση “<?php” και να τελειώνει με την σήμανση “?>” ενώ το αρχείο πρέπει να είναι τύπου “.php”. Έτσι, μας δίνεται η δυνατότητα να τρέξουμε στον εξυπηρετητή μία λειτουργία και να ενσωματώσουμε το αποτέλεσμα στον κώδικα HTML που θα λάβει ο χρήστης. Μία άλλη πολύ σημαντική λειτουργία είναι η χρήση συνόδου. Καθώς η PHP είναι μία γλώσσα σεναρίου, το κάθε σενάριο εκτελείται ξεχωριστά από τα υπόλοιπα. Πολλές φορές όμως χρειαζόμαστε δεδομένα μεταξύ σεναρίων που μπορεί να μεταβάλλονται από αυτά. Αυτή την λειτουργία προσθέτουν οι υπηρεσίες μπισκότων (cookies) και συνεδριών (sessions). Καθώς η υπηρεσία μπισκότων αποθηκεύει τα δεδομένα στο σύστημα του χρήστη, χρησιμοποιείται για συγκεκριμένες λειτουργίες όπως η μνημόνευση των στοιχείων εισόδου του. Η υπηρεσία συνεδριών αντίθετα, αποθηκεύει τα δεδομένα της στο σύστημα του εξυπηρετητή, περιορίζοντας σημαντικά τα θέματα ασφάλειας που μπορούν να προκύψουν από την χρήση της. Έτσι γίνεται φανερό ότι η χρήση συνεδριών μας επιτρέπει την διασύνδεση των σεναρίων με αποτέλεσμα ένα ολοκληρωμένο σύστημα με πολύ μεγάλες δυνατότητες.

```
1  <html>
2      <head>
3          <title>Γειά σου κόσμε - Παράδειγμα</title>
4      </head>
5      <body>
6          <?php echo "Γειά σου κόσμε"; ?>
7      </body>
8  </html>
```

Εικόνα 1: Ένθετος κώδικας PHP σε κώδικα HTML

Στην εφαρμογή που αναπτύξαμε, έγινε αξιοποίηση των παραπάνω δυνατοτήτων σε μεγάλο βαθμό. Ταυτόχρονα όμως δόθηκε βάση στην σωστή οργάνωση του κώδικα που αναπτύχθηκε. Η χρήση της γλώσσας PHP διαχωρίστηκε σε δύο βασικούς τομείς ώστε να δημιουργηθούν αποτελεσματικά οι λειτουργίες που χρειαζόμαστε αλλά ταυτόχρονα να είναι εύκολη και άμεση η διατήρηση και επέκταση της εφαρμογής. Ο πρώτος τομέας βασίζεται στον αντικειμενοστραφή προγραμματισμό. Δημιουργήσαμε κλάσεις που αντιπροσωπεύουν κατηγορίες λειτουργιών που επαναχρησιμοποιούν μέσα στο σύστημα. Οι κλάσεις αυτές βασίζονται σε μεθόδους που εκτελούν συγκεκριμένες λειτουργίες και επιστρέφουν συγκεκριμένες εξόδους. Έτσι η κάθε μέθοδος αντιμετωπίζεται ως ένα μαύρο κουτί που εκτελεί μια συγκεκριμένη λειτουργία, κατανεμημένο στην αντίστοιχη κατηγορία του. Η τεχνική αυτή απλοποιεί σημαντικά τόσο την συντήρηση όσο και την προσθήκη λειτουργιών. Ο δεύτερος τομέας αναλαμβάνει την χρήση του πρώτου. Πρόκειται για τον κώδικα που δεν επαναλαμβάνεται σε ξεχωριστά σενάρια, αλλά προσδιορίζει κάθε φορά τον τρόπο που το σενάριο θα αντιμετωπίσει τις εισόδους που δέχθηκε και το αποτέλεσμα που θα παράγει. Ο τομέας αυτός αναλαμβάνει, μεταξύ άλλων, την φόρτωση των απαραίτητων λειτουργιών, την εξασφάλιση αποστείρωσης των εισαγόμενων δεδομένων, την σωστή αντιμετώπιση αιτημάτων και στοιχείων φόρμας καθώς και την χρήση των αντικειμένων για την δημιουργία και προβολή των ζητούμενων αποτελεσμάτων. Φαίνεται λοιπόν ότι με τον πρώτο τομέα αναπτύξαμε τον σκελετό του συστήματος, ενώ ο δεύτερος τομέας αποτελεί τον ιστό που περιβάλλει το σύστημα και επιτρέπει την ομαλή λειτουργία του.

Η γλώσσα PHP μας παρέχει τελικά ένα ολοκληρωμένο διαδικτυακό σύστημα, εύκολα συντηρήσιμο και επεκτάσιμο. Μας δίνει τα εργαλεία που οδηγούν σε έναν ενεργό εξυπηρετητή, απαραίτητο για τις υπηρεσίες που αποσκοπούμε να προσφέρουμε. Παρόλα αυτά, οι λειτουργίες τόσο του εξυπηρετητή όσο και του περιηγητή δεν είναι αρκετές για τους στόχους που θέσαμε. Καθώς οι λειτουργίες αυτές αλληλοεπιδρούν με τον χρήστη, απαιτείται και μία υπηρεσία που θα δώσει νόημα σε αυτή την αλληλεπίδραση.

2.1.4 MySQL

Η SQL (Structured Query Language, Δομημένη Γλώσσα Διατύπωσης Ερωτήσεων) [18] είναι μία γλώσσα προγραμματισμού, σχεδιασμένη για την αλληλεπίδραση με μία βάση δεδομένων. Η αλληλεπίδραση αυτή γίνεται μέσω συστημάτων διαχείρισης σχεσιακών βάσεων δεδομένων. Ένα από αυτά τα συστήματα είναι η MySQL που χρησιμοποιείται ευρέως στον προγραμματισμό διαδικτύου για την διατήρηση βάσεων δεδομένων, ένα σύστημα μεγάλων επιδόσεων και μικρού κόστους όπως περιγράφεται στο [19].

Η MySQL επιτρέπει στην εφαρμογή μας την αποθήκευση στοιχείων σε μία βάση δεδομένων και την προσπέλασή τους μέσω SQL ερωτημάτων. Η αλληλεπίδραση της εφαρμογής μας με την βάση δεδομένων γίνεται αποκλειστικά μέσω της γλώσσας PHP η οποία μας δίνει την δυνατότητα να θέσουμε ερωτήματα SQL με ασφαλή και άμεσο τρόπο με την χρήση της διεπαφής PDO (PHP Data Objects, PHP Αντικείμενα Δεδομένων) [20]. Καθώς η εφαρμογή μας απαιτεί την χρήση συστήματος χρηστών αλλά και την διατήρηση αρχείων, έργων και εξαρτημάτων, γίνεται φανερό ότι πρέπει να διατηρήσουμε κάποια δεδομένα. Έτσι χρησιμοποιούμε την βάση δεδομένων για την αποθήκευση στοιχείων χωρισμένα σε κατηγορίες που αντιστοιχούν σε διαφορετικούς πίνακες βάσης δεδομένων. Σε αυτούς τους πίνακες διατηρούμε δεδομένα όπως τις λεπτομέρειες που προσδιορίζουν τον κάθε χρήστη, τα στοιχεία κάθε προγράμματος και αρχείου καθώς και αυτά κάθε δημοσιευμένου εξαρτήματος. Παράλληλα όμως αποθηκεύουμε και συνδετικά στοιχεία όπως τους συγγραφείς που αντιστοιχούν σε κάθε έργο και εγγραφές που συνδέουν δύο πίνακες. Μέσω της MySQL μπορούμε να προσθέσουμε, τροποποιήσουμε, αφαιρέσουμε και προβάλουμε αυτές τις πληροφορίες. Έτσι η MySQL μας επιτρέπει την διατήρηση μνήμης στο σύστημά μας και την ομαλή συνεχή ροή πληροφοριών μεταξύ χρηστών και εφαρμογής.

Με τα παραπάνω εργαλεία δημιουργήσαμε τις βασικές λειτουργίες της εφαρμογής μας. Μία εφαρμογή που αποτελείται από έναν συνεχή κύκλο αλληλεπίδρασης μεταξύ λειτουργιών εξυπηρετητή και περιηγητή συνδέοντας τελικά την μνήμη του συστήματος με τις απαιτήσεις του χρήστη.

2.1.5 Πρωτόκολλο HTTP

Το HTTP (Hypertext Transfer Protocol , πρωτόκολλο μεταφοράς υπερκειμένου) [21] είναι ένα πρωτόκολλο που αποτελεί ριζικό σημείο αναφοράς για την λειτουργία του διαδικτύου. Έτσι αφού είδαμε τις γλώσσες προγραμματισμού που χρησιμοποιήσαμε, αξίζει να αναφερθούμε σε αυτό το πρωτόκολλο καθώς εισάγει έννοιες που χρησιμοποιούμε ευρέως στην ανάλυση της εφαρμογής.

Οι βασικές έννοιες που καθορίζονται από το HTTP είναι το μοντέλο πελάτη - διακομιστή και η λειτουργία αιτήματος και ανταπόκρισης. Η οντότητα του πελάτη δημιουργεί αιτήματα στην οντότητα του διακομιστή. Ο διακομιστής επεξεργάζεται το αίτημα και επιστρέφει μία ανταπόκριση περιέχοντας πληροφορίες για την κατάσταση της αίτησης και συνήθως δεδομένα που ζητήθηκαν. Στο διαδίκτυο, η οντότητα του πελάτη είναι ο περιηγητής του χρήστη και η οντότητα του διακομιστή είναι ο εξυπηρετητής. Τα συνηθέστερα αιτήματα είναι αυτά για την λήψη των στοιχείων μίας σελίδας όπου λαμβάνουν ως ανταπόκριση τα δεδομένα που χρειάζεται ο περιηγητής για την εμφάνιση της σελίδας. Τα διαθέσιμα αιτήματα που παρέχει το HTTP χωρίζονται σε μεθόδους. Οι μέθοδοι περιγράφουν τον τρόπο με τον οποίο θα αντιμετωπιστούν τα αιτήματα από τον διακομιστή και το είδος των αναμενόμενων αποτελεσμάτων από τον πελάτη. Οι μέθοδοι που χρησιμοποιούμε στην εφαρμογή μας και αξίζει να δούμε ποιο αναλυτικά είναι οι POST και GET. Η μέθοδος POST χρησιμοποιείται σε αιτήματα που περιέχουν δεδομένα για επεξεργασία από τον διακομιστή. Η μέθοδος αυτή βρίσκεται συνήθως στην αποστολή μίας φόρμας ενώ μπορεί να επιλεχθεί κατά την αποστολή δεδομένων μέσω AJAX. Συγκεκριμένα η PHP λαμβάνει τα δεδομένα της μεθόδου POST σε

έναν πίνακα με την ονομασία "$_POST". Έτσι εάν ο πελάτης κάνει αποστολή μίας φόρμας με ένα στοιχείο ονομασμένο "variable" που περιέχει την τιμή "value", κατά την εκτέλεση του σεναρίου PHP στον εξυπηρετητή θα έχουμε πρόσβαση στην τιμή του πίνακα "$_POST['variable']=='value'". Η μέθοδος GET αντίθετα χρησιμοποιείται για την ανάκτηση πληροφορίας από την διακομιστή. Η χρήση της μεθόδου GET γίνεται συνήθως μέσο του συνδέσμου αλλά μπορεί και αυτή η μέθοδος επιλεχθεί κατά την χρήση AJAX. Για την περίπτωση περιεχόμενων δεδομένων στην μέθοδο GET η PHP χρησιμοποιεί τον πίνακα "$_GET". Έτσι μεταβαίνοντας στον θεωρητικό σύνδεσμο "http://example.com/index.php?var=value" το σενάριο PHP στον εξυπηρετητή θα περιέχει την τιμή στον πίνακα "$_GET['var']=='value'". Βλέπουμε λοιπόν ότι έχοντας έναν λειτουργικό εξυπηρετητή και δίνοντας την δυνατότητα διάδραση του χρήστη με την εφαρμογή, οι παραπάνω μέθοδοι βρίσκουν εφαρμογή σε πολλές λειτουργίες που χρησιμοποιούμε.

2.2 Εξωτερικά Εργαλεία που Χρησιμοποιήθηκαν

Τα βασικά εργαλεία που παρουσιάστηκαν παραπάνω αποτελούν μέρος του προγραμματισμού διαδικτύου για δεκαετίες. Ως αποτέλεσμα αυτού του γεγονότος, υπάρχουν πολλά βοηθητικά εργαλεία που έχουν αναπτυχθεί με σκοπό την προσθήκη ενός περαιτέρω αφαιρετικού επιπέδου. Τέτοια εργαλεία ποικίλουν από εξιδεικευμένες λειτουργίες ως βιβλιοθήκες και πλαίσια. Στις επόμενες ενότητες παρουσιάζονται τα εξωτερικά εργαλεία αυτού του είδους που χρησιμοποιήθηκαν για την εφαρμογή μας, προσθέτοντας σύγχρονες και δοκιμασμένες λύσεις για τις λειτουργίες που χρειαζόμαστε.

2.2.1 Πλαίσιο Bootstrap

Το Bootstrap [22] είναι ένα HTML, CSS και JavaScript πλαίσιο. Παρέχει έτοιμες CSS κλάσεις γενικής χρήσης και JavaScript γεγονότα πάνω σε κάποιες από αυτές τις κλάσεις για συνήθεις λειτουργίες. Μας δίνει την δυνατότητα να βασίσουμε τον βασικό σκελετό του θέματος εμφάνισης που αναπτύξαμε πάνω σε ένα δοκιμασμένο και έμπειρο σύστημα. Το πλαίσιο Bootstrap που χρησιμοποιούμε είναι η $3^{η}$ έκδοση και συγκεκριμένα η έκδοση Bootstrap 3.3.6.

Οι δυνατότητες που προσφέρει το Bootstrap είναι πολλές αλλά υπάρχουν κάποιες από αυτές που αξίζει να αναφερθούν ξεχωριστά. Η πρώτη δυνατότητά του, που μας οδήγησε και στη επιλογή αυτού του εργαλείου, είναι ότι ανταποκρίνεται σωστά στις διαστάσεις της οθόνης του χρήστη. Καθώς θεωρούμε ότι είναι σημαντικό για την εφαρμογή μας να προσφέρει σε κάθε χρήστη την δυνατότητα προσπέλασης της από ποικιλία συσκευών, είναι φανερό γιατί μας ενδιαφέρει σε μεγάλο βαθμό μια τέτοια δυνατότητα. Μία ακόμη δυνατότητα του Bootstrap είναι η χρήση πλέγματος. Οι βασικές κλάσεις που χρησιμοποιεί το Bootstrap για την δημιουργία μίας ιστοσελίδας προτείνουν τον διαχωρισμό των στοιχείων σε σειρές και στήλες δημιουργώντας ένα πλέγμα. Εκτός από την πολύ καλή συνεργία αυτής της τεχνικής με την πρώτη δυνατότητα που αναφέραμε, η χρήση πλέγματος οδηγεί ταυτόχρονα σε μια πολύ

οργανωμένη εμφανισιακά ιστοσελίδα που προσφέρει στον χρήστη τόσο την καλαισθησία που επιθυμούμε όσο και την ευκολία κατανόησης και εύρεσης των επιθυμιών λειτουργιών. Τέλος, το Bootstrap μας προσφέρει ένα σύνολο κλάσεων και JavaScript λειτουργιών που χρησιμοποιούνται ευρέως και μας απαλλάσσουν από τον χρόνο επαναδημιουργίας τους αλλά ταυτόχρονα προσφέρονται με οργανωμένο τρόπο ώστε να μας επιτρέπουν την περαιτέρω επέκτασή και εξειδίκευσή τους στις δικές μας ανάγκες. Τέτοιες λειτουργίες ποικίλουν από την δημιουργία μπάρας πλοήγησης με υπομενού ως την σωστή στοίχιση στοιχείων και την επαλήθευση του τύπου δεδομένων σε μία φόρμα αποστολής δεδομένων. Η χρήση του πλαισίου Bootstrap στην εφαρμογή που αναπτύξαμε μας έδωσε τις βασικές λειτουργίες που χρειαστήκαμε για την δημιουργία ενός θέματος εμφάνισης ενώ παράλληλα μας επέτρεψε την εύκολη και άμεση τροποποίηση και εφαρμογή τους στις δικές μας απαιτήσεις.

2.2.2 Βιβλιοθήκη jQuery

Η jQuery [23] είναι μία πολύ διαδεδομένη βιβλιοθήκη για την γλώσσα JavaScript. Η βιβλιοθήκη αυτή μας προσφέρει ένα αφαιρετικό επίπεδο απλοποιώντας πολλές JavaScript λειτουργίες. Συγκεκριμένα στην εφαρμογή μας, αξιοποιούμε την έκδοση jQuery 2.2 καθώς αυτή απαιτείται από το Bootstrap 3.

Εισάγοντας την βιβλιοθήκη jQuery στην εφαρμογή μας, χρησιμοποιούμε την γλώσσα JavaScript μέσα από την jQuery. Οι βασικές λειτουργίες παραμένουν οι ίδιες αλλά αποφεύγεται πλέον η χρήση απλών JavaScript συναρτήσεων, αντικειμένων και μεθόδων. Η τεχνική αυτή δεν είναι απαραίτητη αλλά αποσκοπεί σε καθαρότερο κώδικα και την αποφυγή χρήσης μη συμβατών τύπων δεδομένων ως είσοδο ή έξοδο συναρτήσεων. Έτσι πλέον ορίζουμε γεγονότα δεύτερου επιπέδου μέσω της jQuery, επιλέγουμε στοιχεία HTML ως αντικείμενα jQuery και χρησιμοποιούμε σε αυτά τις μεθόδους που θέτονται από την βιβλιοθήκη jQuery. Το αφαιρετικό επίπεδο που εισάγεται από την βιβλιοθήκη στοχεύει τόσο στην άμεση πρόσβαση σε λειτουργίες που δίνονται από την απλή JavaScript όσο και στην εισαγωγή χρήσιμων λειτουργιών που συχνά ο προγραμματιστής απλής JavaScript θα πρέπει να αναδημιουργεί κάθε φορά. Χρησιμοποιώντας λοιπόν την βιβλιοθήκη jQuery, αναπτύξαμε τις απαιτούμενες λειτουργίες JavaScript σε καθαρή και εύκολα συντηρήσιμη μορφή χωρίς να χρειαστούν συμβιβασμοί στους στόχους που θέσαμε.

2.2.3 Κειμενογράφος Ace

Ο Ace [24] είναι ένας κειμενογράφος ανεπτυγμένος σε JavaScript. Έχει εξειδικευτεί για την χρήση του στην συγγραφή κώδικα με επισημάνσεις συντακτικού για πάνω από 110 γλώσσες προγραμματισμού.

Η εφαρμογή που αναπτύξαμε στοχεύει, μεταξύ άλλων, στην συγγραφή του κώδικα. Έτσι η χρήση ενός κειμενογράφου γίνεται μια απαραίτητη προϋπόθεση για την σωστή διεπαφή του χρήστη με τον κώδικά του. Ο κειμενογράφος Ace επιλέχθηκε τόσο λόγο του τρόπου διανομής όσο και των δυνατοτήτων που προσφέρει. Ακολουθώντας την πολιτική που ορίσαμε στους στόχους μας, χρησιμοποιούμε των Ace καθώς προσφέρεται δωρεάν τόσο το ίδιο το

πρόγραμμα, όσο και ο ανοικτός κώδικάς του. Το γεγονός αυτό αποδείχθηκε τελικά απαραίτητο όταν εμφανίστηκαν συμβατικές αδυναμίες μεταξύ Ace και Bootstrap όπου συγκεκριμένες τροποποιήσεις στον κώδικα του Ace μας επέτρεψαν να τις ξεπεράσουμε. Παρόλο του τρόπου διανομής του όμως, ο κειμενογράφος Ace παραμένει ένα πολύ δυνατό εργαλείο με χρήσιμες λειτουργίες. Προσφέρει επισημάνσεις για τις γλώσσες που μας ενδιαφέρουν, πάνω από 20 θέματα εμφάνισης, αρίθμηση γραμμών, αναδίπλωση γραμμών και κώδικα, εύκολη μεταφορά κώδικα από τοπικά αρχεία με επιλογή, μεταφορά και απόθεση κειμένου με χρήση ποντικιού καθώς και πλήθος άλλων λειτουργιών. Η χρήση τελικά αυτού του κειμενογράφου μας εφοδίασε με ένα δυνατό εργαλείο που ενσωματώσαμε αποτελεσματικά στην εφαρμογή μας και προσαρμόσαμε χωρίς δυσκολίες στους στόχους μας.

2.2.4 GHDL

Το εργαλείο GHDL [25] αναλαμβάνει την μεταγλώττιση και προσομοίωση της γλώσσας VHDL. Το GHDL μεταγλωττίζει τον κώδικα VHDL σε γλώσσα μηχανής δίνοντας την δυνατότητα για γρηγορότερη προσομοίωση από εργαλεία που τον μετατρέπουν σε κάποια ενδιάμεση γλώσσα προγραμματισμού. Ενσωματώσαμε λοιπόν το εργαλείο αυτό στην εφαρμογή μας με σκοπό την άμεση και γρήγορη εξαγωγή αποτελεσμάτων προσομοίωσης.

Το GHDL είναι ένα τοπικό εργαλείο. Το εργαλείο αυτό χρησιμοποιείται από τον εξυπηρετητή προσφέροντας τις δυνατότητές του στον χρήστη χωρίς την ανάγκη μεταφόρτωσης και εγκατάστασής του. Στο σύστημά μας, διατηρούμε έναν φάκελο αρχείων εργασίας. Ένα σενάριο τερματικού ελέγχει συνεχώς αυτόν τον φάκελο για αίτηση εργασίας. Όταν ένας χρήστης θέλει να μεταγλωττίσει ένα αρχείο ή να προσομοιώσει ένα πρόγραμμα, δημιουργείται ένα αρχείο αίτησης εργασίας με τις απαραίτητες εντολές για την αντίστοιχη μεταγλώττιση ή προσομοίωση. Τότε το σενάριο τερματικού θα εκτελέσει την λειτουργία, εκτελώντας το εργαλείο GHDL με τις κατάλληλες εισόδους, και θα απομακρύνει το αρχείο εργασίας. Έτσι, έχουμε ένα δυνατό και γρήγορο τοπικό εργαλείο, προσαρμοσμένο με κατάλληλο τρόπο ώστε να μεταφέρονται αποτελεσματικά οι δυνατότητές του στην δική μας εφαρμογή, σε ένα διαδικτυακό περιβάλλον.

2.2 Σύνοψη Κεφαλαίου

Προσθέτοντας τα εξωτερικά εργαλεία ενός ανεπτυγμένου συστήματος με τα βασικά εργαλεία προγραμματισμού διαδικτύου που παρουσιάστηκαν, έχουμε τα δομικά υλικά της εφαρμογής μας. Χρησιμοποιώντας αυτά τα δομικά υλικά, αναπτύξαμε μία εφαρμογή που καλύπτει αποτελεσματικά τους στόχους που θέσαμε στα πλαίσια της πολιτικής ελεύθερου λογισμικού και ανοιχτού κώδικα. Θα αναλύσουμε παρακάτω τον τρόπο που αξιοποιήθηκαν τα παραπάνω εργαλεία για την σύνθεση των λειτουργιών που αποτελούν την εφαρμογή που αναπτύξαμε.

Κεφάλαιο 3 - Ανάπτυξη της Εφαρμογής

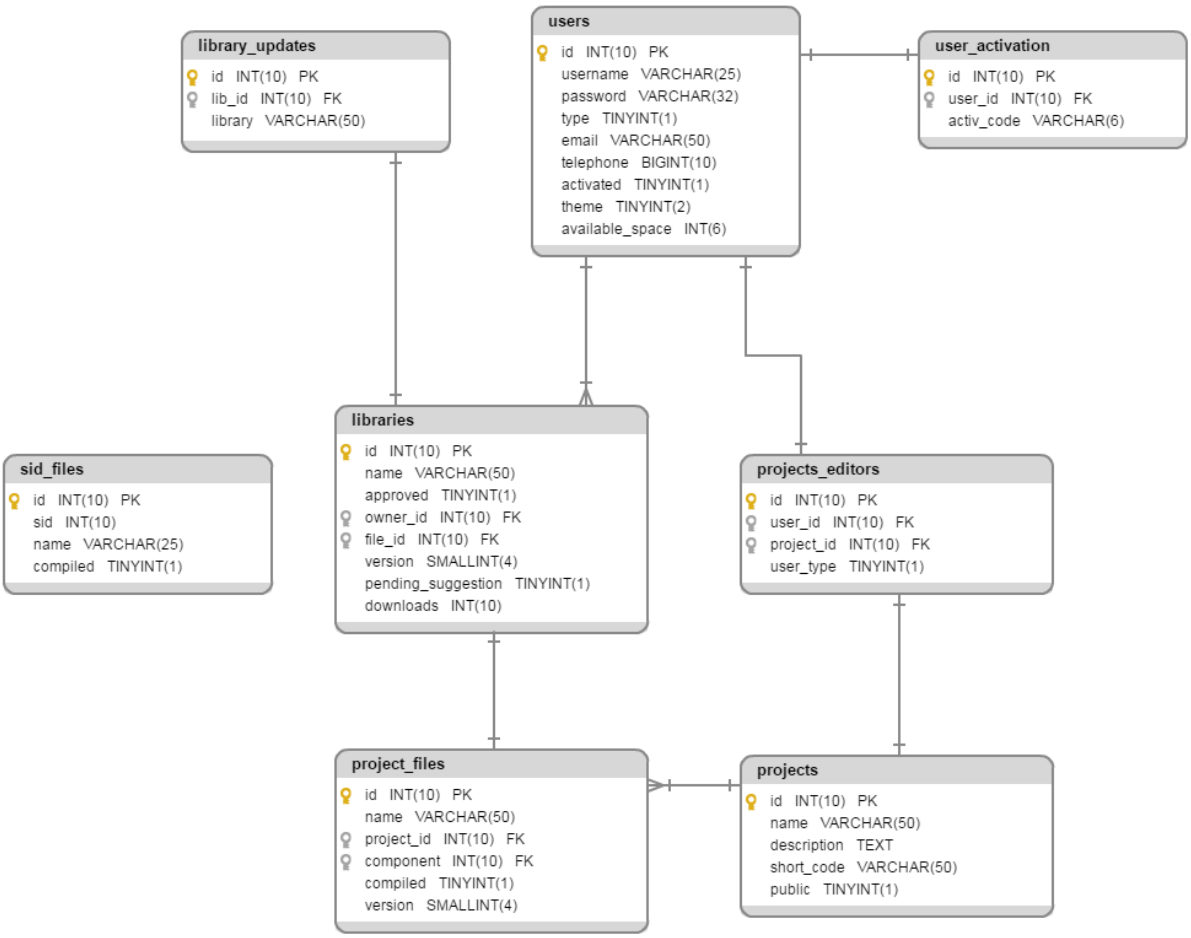
Στα προηγούμενα κεφάλαια αναπτύξαμε τόσο τους στόχους και τα κίνητρα για την υλοποίηση της εφαρμογής μας, όσο και τα δομικά υλικά που χρησιμοποιήσαμε. Σε αυτό το κεφάλαιο γίνεται μία αναλυτική περιγραφή του τρόπου που χρησιμοποιούμε τα βασικά και εξωτερικά εργαλεία για την υλοποίηση των λειτουργιών που συνθέτουν την εφαρμογή για την κάλυψη των στόχων μας. Συγκεκριμένα, θα αναλύσουμε αρχικά την διάρθρωση της βάσης δεδομένων και των απαιτούμενων καταλόγων αρχείων, θα προχωρήσουμε σε μία λεπτομερή εξέταση στον τρόπο ανάπτυξης και υλοποίησης των λειτουργιών που συνθέτουν την εφαρμογή μας και το εργαλείο σχεδίασης κυματομορφών και τέλος θα δώσουμε τα απαραίτητα βήματα για την εγκατάσταση της εφαρμογής που αναπτύξαμε σε έναν εξυπηρετητή.

3.1 Διάρθρωση της Βάσης Δεδομένων

Η σχεδίαση της βάσης δεδομένων είναι συνήθως ένα από τα πρώτα βήματα υλοποίησης μίας εφαρμογής. Κατά την σχεδίαση αυτή λαμβάνονται υπόψη πολλά θέματα για τον τρόπο λειτουργίας και τις λεπτομέρειες υλοποίησης. Η σωστή σχεδίαση μίας βάσης δεδομένων αποσκοπεί στον διαχωρισμό δεδομένων σε πίνακες με ευκρινή σκοπό. Συγκεκριμένα σε μία εφαρμογή διαδικτύου, είναι σημαντικό να σχεδιάσουμε την βάση με τέτοιο τρόπο ώστε να βελτιστοποιήσουμε την δυνατότητα λήψης και τροποποίησης δεδομένων με τις ελάχιστες δυνατές ερωτήσεις.

Έχοντας υπόψη τα παραπάνω, διαχωρίσαμε την βάση δεδομένων μας σε εφτά πίνακες. Ο κάθε πίνακας έχει μία ξεχωριστή κατηγορία αλλά ταυτόχρονα υπάρχουν διακριτοί σύνδεσμοι μεταξύ τους. Γενικά, υπάρχει ένα μοτίβο στον τρόπου που χρησιμοποιούμε τους πίνακες της βάσης δεδομένων μας. Ο κάθε πίνακας περιέχει ένα γνώρισμα ταυτοποίησης που ονομάζουμε "id". Το γνώρισμα αυτό λειτουργεί ως πρωτεύον κλειδί και παίρνει τιμές με αυτόματη αύξηση σε κάθε καταχώριση. Εάν υπάρχει σύνδεσμος με άλλον πίνακα, χρησιμοποιούμε ένα γνώρισμα που αντιστοιχεί στο "id" του άλλου πίνακα. Η λογική που ακολουθούμε είναι αυτή του ξένου κλειδιού. Μία εφαρμογή που βρίσκεται σε πολλές βάσεις δεδομένων για την δημιουργία συνδέσμων μεταξύ πινάκων. Παρόλα αυτά, δεν χρησιμοποιούμε ξένα κλειδιά στην ίδια την βάση καθώς κάτι τέτοιο θέτει περιορισμούς στο είδος της μηχανής βάσης δεδομένων που μπορεί να χρησιμοποιήσει η εφαρμογή μας. Η επιρροή αυτής της απόφασης είναι η ανάγκη για έλεγχο κατά την κατάργηση εγγραφών για εξαρτώμενες εγγραφές σε άλλους πίνακες αλλά και η απεξάρτηση της βάσης δεδομένων μας από συγκεκριμένες τεχνολογίες. Καθώς χρησιμοποιούμε μία σωστή σχεδίαση βάσης δεδομένων, ο έλεγχος κατάργησης εξαρτώμενων τιμών απλοποιείται σημαντικά ενώ λόγο της φύσης της εφαρμογής έχουμε πολύ μικρό αναμενόμενο πλήθος τέτοιων ερωτημάτων. Αντίθετα, η ικανότητα μεταφοράς της εφαρμογής σε πλήθος εξυπηρετητών με υποστήριξη διαφορετικών τεχνολογιών επιτρέπει σε μεγαλύτερο πλήθος ενδιαφερόμενων την δοκιμή και πιθανώς υποστήριξη και μελλοντική επέκταση της. Έτσι γίνεται φανερό ότι τα πλεονεκτήματα που μας επιφέρει αυτή η επιλογή υπερτερούν των μειονεκτημάτων.

Οι πίνακες που χρησιμοποιήσαμε φαίνονται στην Εικόνα 2 όπως και μεταξύ τους διασυνδέσεις. Φαίνεται όπως εξηγήσαμε παραπάνω το "id" ως PK (πρωτεύον κλειδί) και ως FK (ξένο κλειδί) τα γνωρίσματα διασύνδεσης. Οι βασικές κατηγορίες που αντιπροσωπεύουν οι πίνακες είναι οι χρήστες του συστήματος, τα έργα ή προγράμματα, οι χρήστες με δικαιώματα επεξεργασίας κάθε έργου, τα αρχεία κάθε έργου, τα αρχεία επισκεπτών, οι βιβλιοθήκες ή εξαρτήματα και τέλος τα αρχεία ενημέρωσης βιβλιοθηκών ή εξαρτημάτων. Παρακάτω θα εξηγήσουμε αναλυτικά τον εσωτερικό σχεδιασμό κάθε πίνακα καθώς και τον τρόπο που χρησιμοποιείται από την εφαρμογή μας.



Εικόνα 2: Η δομή της βάσης δεδομένων

3.1.1 Περιγραφή του Πίνακα users

Ο πίνακας users χρησιμοποιείται για την αποθήκευση των δεδομένων κάθε λογαριασμού χρήστη. Η κάθε εγγραφή στον πίνακα αυτό αντιπροσωπεύει έναν ξεχωριστό λογαριασμό. Ο πίνακας αυτός είναι απαραίτητος για την ύπαρξη ενός ολοκληρωμένου συστήματος χρηστών και αποτελείται από ένα σύνολο γνωρισμάτων τόσο για την επιτέλεση αναγκαίων λειτουργιών, όσο για την ταυτοποίηση αλλά και επικοινωνία με τον χρήστη. Στην συνέχεια θα αναλύσουμε το κάθε γνώρισμα ξεχωριστά όπως παρουσιάζονται στην Εικόνα 3.

Name	Type	Collation	Attributes	Null	Default	Extra
id	int(10)			No	*None*	AUTO_INCREMENT
username	varchar(25)	utf8_general_ci		No	*None*	
password	varchar(32)	utf8_general_ci		No	*None*	
type	tinyint(1)			No	0	
email	varchar(50)	utf8_general_ci		No	*None*	
telephone	bigint(10)			Yes	*NULL*	
activated	tinyint(1)			No	0	
theme	tinyint(2)			No	0	
available_space	int(6)			No	20	

Εικόνα 3: Η δομή του πίνακα users

Αρχικά έχουμε το γνώρισμα ταυτοποίησης "id". Αυτό το γνώρισμα παίρνει μοναδική τιμή για κάθε εγγραφή και χρησιμοποιείται ως πρωτεύον κλειδί. Ο τύπος του γνωρίσματος "id" είναι "int(10)", δηλαδή ακέραιος αριθμός μέχρι 10 ψηφία. Ένα γνώρισμα "int" αντιπροσωπεύεται με 4 Byte, έτσι έχει ως μέγιστη τιμή την 2,147,483,647 και ως ελάχιστη την -2,147,483,648. Το επόμενο γνώρισμα του πίνακα είναι το "username". Το γνώρισμα αυτό είναι τύπου "varchar(25)", μεταβλητό πεδίο χαρακτήρων με μέγιστο μέγεθος 25 χαρακτήρων. Χρησιμοποιούμε κωδικοποίηση utf8_general_ci σε όλα τα γνωρίσματα "varchar" για την σωστή καταχώριση ελληνικών χαρακτήρων. Έπειτα έχουμε το γνώρισμα "password". Το γνώρισμα αυτό χρησιμοποιείται για την αποθήκευση του κωδικού χρήστη αφού τον περάσουμε από τον αλγόριθμο κατατεμαχισμού MD5 [26]. Αν και υπάρχουν διάφοροι αλγόριθμοι, το MD5 είναι υπεραρκετό για την ασφάλεια που θέλουμε. Έτσι χρησιμοποιούμε "varchar(32)" καθώς η έξοδος του αλγορίθμου MD5 αποτελείται από 32 ψηφία συνδυασμού αριθμών και χαρακτήρων. Το επόμενο γνώρισμα "type" είναι τύπου "tinyint(1)". Ο τύπος αυτός αποθηκεύει ακέραιες μεταβλητές με χρήση ενός Byte επιτρέποντας τιμές από 0 ως 255. Το γνώρισμα "type" αντιστοιχεί στον τύπο χρήστη που αντιπροσωπεύει δικαιώματα και λειτουργίες χρήστη με την τιμή 0 ή διαχειριστή με την τιμή 1. Το γνώρισμα "email" είναι τύπου "varchar(50)" επιτρέποντας την συγκράτηση της διεύθυνσης e-mail του χρήστη. Το μέγεθος 50 χαρακτήρων είναι μία χρυσή τομή καθώς οι περισσότερες έγκυρες διευθύνσεις είναι συνήθως κάτω από 50 χαρακτήρες. Το γνώρισμα "telephone" χρησιμοποιείται προαιρετικά για την συγκράτηση ενός τηλεφωνικού αριθμού. Καθώς οι αριθμοί αυτοί είναι 10 ψηφίων αλλά και μεγαλύτεροι του 2,147,483,647, επιλέξαμε την χρήση τύπου "bigint(10)". Ο τύπος "bigint" μας επιτρέπει την χρήση 8 Bytes για την αποθήκευση τιμών καλύπτοντας εύκολα τις ανάγκες συγκράτησης ενός αριθμού 10 ψηφίων. Καθώς το γνώρισμα αυτό είναι προαιρετικό, επιτρέπεται η χρήση κενού σε περίπτωση που δεν χρησιμοποιείται. Το γνώρισμα "activated" είναι τύπου "tinyint(1)" καθώς χρησιμοποιείται ως σημαία επισήμανσης. Η επισήμανση που προκύπτει από την τιμή 0 είναι ότι ο χρήστης δεν έχει επιβεβαιώσει την διεύθυνση e-mail του, ενώ η τιμή 1 επισημάνει ότι ο χρήστης έχει επιβεβαιώσει επιτυχώς την διεύθυνση e-mail που έδωσε στο σύστημα. Το γνώρισμα "theme" είναι τύπου "tinyint(2)". Το συγκεκριμένο γνώρισμα συγκρατεί το επιλεγμένο θέμα εμφάνισης για τον επεξεργαστή κειμένου. Καθώς υπάρχουν 24 διαθέσιμα

θέματα, ο τύπος “tinyint” με 2 ψηφία μπορεί εύκολα να καλύψει τις απαραίτητες τιμές. Τέλος, το γνώρισμα “available_space” είναι τύπου “int(6)” και αντιπροσωπεύει τον διαθέσιμο χώρο στον εξυπηρετητή για κάθε χρήστη σε MegaBytes. Ο πίνακας “users” λοιπόν, αποτελείται από τα γνωρίσματα που συντελούν ένα λογαριασμό χρήστη. Τα γνωρίσματα αυτά αντιπροσωπεύουν τόσο τις τιμές απαραίτητες για διάφορες λειτουργίες όσο και τις τιμές που επιφέρουν πληροφορίες για τον χρήστη.

3.1.2 Περιγραφή του Πίνακα user_activation

Ο πίνακας “user_activation” είναι υπεύθυνος για την διατήρηση των κωδικών ενεργοποίησης κάθε λογαριασμού χρήστη. Καθώς η ενεργοποίηση γίνεται μία φορά για κάθε λογαριασμό, μπορούμε να αφαιρέσουμε την εγγραφή από τον πίνακα κατά την διαδικασία της. Έτσι χρησιμοποιούμε ξεχωριστό πίνακα από αυτόν των χρηστών για την διατήρηση των σχετικών στοιχείων αυτής της λειτουργίας.

#	Name	Type	Collation	Attributes	Null	Default	Extra
1	**id** 🔑	int(10)			No	*None*	AUTO_INCREMENT
2	**user_id**	int(10)			No	*None*	
3	**activ_code**	varchar(6)	utf8_general_ci		No	*None*	

Εικόνα 4: Η δομή του πίνακα user_activation

Το πρώτο γνώρισμα του πίνακα είναι το γνώρισμα ταυτοποίησης “id”. Το γνώρισμα αυτό αποτελεί το πρωτεύον κλειδί του πίνακα και είναι τύπου “int(10)” με αυτόματη αύξηση τιμής. Το δεύτερο γνώρισμα του πίνακα είναι το “user_id”. Το γνώρισμα αυτό είναι τύπου “int(10)” και αποτελεί ξένο κλειδί συνδέοντας αυτόν τον πίνακα με το πρωτεύον κλειδί του πίνακα χρηστών. Τέλος, το γνώρισμα “active_code” είναι τύπου “varchar(6)” και αντιπροσωπεύει έναν τυχαίο εξαψήφιο κωδικό, συνδυασμού των μονοψήφιων αριθμών και κεφαλαίων λατινικών χαρακτήρων, που χρησιμοποιείται για την επιβεβαίωση της διεύθυνσης e-mail του κάθε χρήστη. Δεδομένου ότι ο κωδικός αυτός χρειάζεται να είναι μόνο τυχαίος και όχι μοναδικός για κάθε χρήστη, τα έξι ψηφία σε συνδυασμό με τις απαραίτητες καθυστερήσεις κάθε δοκιμής καθιστούν το συγκεκριμένο μέγεθος αρκετό για μία καλή εξασφάλιση της σωστής λειτουργίας του και της δυσκολίας παράκαμψής του.

3.1.2 Περιγραφή του Πίνακα projects

Ο πίνακας “projects” περιέχει τα γνωρίσματα που περιγράφουν το κάθε έργο ή πρόγραμμα. Η κάθε εγγραφή περιγράφει ένα διαφορετικό έργο. Υπήρξε η επιλογή για την αποφυγή δημιουργίας αυτού του πίνακα καθώς θα μπορούσαμε να επιτρέψουμε μόνο ένα έργο σε κάθε χρήστη ή την χρήση καταλόγων και υποκαταλόγων τα οποία θα μπορούσε ο χρήστης να θεωρήσει ως διαφορετικά έργα. Χρησιμοποιώντας όμως αυτόν τον πίνακα σε συνδυασμό με τους υπόλοιπους, ανοίγουμε τον δρόμο για ένα σύνολο λειτουργιών που διαφορετικά δεν θα

ήταν προσιτές. Συγκεκριμένα επιτρέπουμε την εμφάνιση των διαφορετικών έργων με τον τρόπο που επιθυμούμε, την επίδειξη ή απόκρυψη αυτών από άλλους χρήστες καθώς και την προσθήκη περιγραφής για μία πιο λεπτομερής ανάλυση του σκοπού κάθε έργου. Παρακάτω εξηγούμε το κάθε γνώρισμα ξεχωριστά όπως αυτά φαίνονται στην Εικόνα 5.

Name	Type	Collation	Attributes	Null	Default	Extra
id	int(10)			No	*None*	AUTO_INCREMENT
name	varchar(50)	utf8_general_ci		No	*None*	
description	text	utf8_general_ci		No	*None*	
short_code	varchar(50)	utf8_general_ci		No	*None*	
public	tinyint(1)			No	1	

Εικόνα 5: Η δομή του πίνακα projects

Όπως κάθε πίνακας, έτσι και ο "projects" έχει αρχικά ένα γνώρισμα ταυτοποίησης "id". Το γνώρισμα αυτό έχει μοναδικές τιμές για κάθε εγγραφή του πίνακα καθώς η τιμή του αυξάνεται αυτόματα και χρησιμοποιείται ως πρωτεύον κλειδί. Το γνώρισμα "name" είναι τύπου "varchar(50)" και αντιπροσωπεύει τον τίτλο του έργου. Έχοντας διαθέσιμους 50 χαρακτήρες επιτρέπουμε έναν περιεκτικό αλλά όχι υπερβολικά περιορισμένο τίτλο. Το γνώρισμα "description" είναι τύπου "text" και περιέχει την περιγραφή του έργου. Ο τύπος "text" επιτρέπει την αποθήκευση κειμένου έως 255 χαρακτήρων, μέγεθος που θεωρούμε ικανοποιητικό για την σύντομη περιγραφή ενός έργου. Το επόμενο γνώρισμα είναι το "short_code", τύπου "varchar(50)". Το συγκεκριμένο γνώρισμα χρησιμοποιείται για την συγκράτηση του ονόματος καταλόγου που περιέχει το συγκεκριμένο έργο. Το "short_code" προκύπτει από τον τίτλο μεταγλωττίζοντας τους ελληνικούς χαρακτήρες σε αντίστοιχους λατινικούς και τα κενά σε κάτω παύλες, έτσι βεβαιώνουμε ότι το σύστημα του εξυπηρετητή θα μπορέσει να δημιουργήσει και προσπελάσει τον αντίστοιχο κατάλογο. Τέλος, έχουμε το γνώρισμα "public", τύπου "tinyint(1)". Το γνώρισμα αυτό λειτουργεί ως σημαία σήμανσης με την τιμή 1 να σημαίνει ότι το έργο μπορεί να προσπελαστεί από άλλους χρήστες και την τιμή 0 ότι μπορεί να προσπελαστεί μόνο από τους συγγραφείς και διαχειριστές. Έτσι διατηρούμε στον πίνακα "projects" τις απαραίτητες πληροφορίες για κάθε έργο. Όπως θα δούμε παρακάτω όμως, ένα έργο έχει νόημα μόνο όταν έχει περιεχόμενο και συγγραφέα.

3.1.3 Περιγραφή του Πίνακα project_files

Ο πίνακας "project_files" περιέχει τα γνωρίσματα που περιγράφουν το κάθε αρχείο μέσα σε κάθε έργο. Παρόμοια με την περίπτωση του πίνακα "projects" υπήρξε η επιλογή να χρησιμοποιήσουμε κλήσεις συστήματος για να βρούμε τα αρχεία κάθε έργου. Χρησιμοποιώντας όμως τον πίνακα "project_files" αποκτούμε πρόσβαση σε λειτουργίες όπως η σήμανση ανάγκης για επανάληψη μεταγλώττισης του αρχείου μετά από αλλαγές καθώς και την σύνδεσή του με το αντίστοιχο εξάρτημα και ενημέρωση για ύπαρξη καινούριας έκδοσης. Περιγράφουμε στην συνέχεια τα γνωρίσματα του πίνακα όπως παρουσιάζονται στην Εικόνα 6.

Name	Type	Collation	Attributes	Null	Default	Extra
id	int(10)			No	*None*	AUTO_INCREMENT
name	varchar(50)	utf8_general_ci		No	*None*	
project_id	int(10)			No	*None*	
compiled	tinyint(1)			No	0	
component	int(10)			No	0	
version	smallint(4)			No	0	

Εικόνα 6: Η δομή του πίνακα project_files

Το πρωτεύον κλειδί που χρησιμοποιούμε κατά κανόνα εμφανίζεται και στον πίνακα "project_files" ως το γνώρισμα "id". Το δεύτερο γνώρισμα είναι το "name", τύπου "varchar(50)". Το γνώρισμα αυτό συγκρατεί το όνομα του αρχείου με την κατάληξή του. Αν και είναι σπάνιο να χρειαστούν 50 χαρακτήρες για την ονομασία ενός αρχείου, είναι πιθανό σε εξαρτήματα με ονομασία που περιγράφει την λειτουργία τους. Το επόμενο γνώρισμα είναι το "project_id". Το γνώρισμα αυτό είναι τύπου "int(10)" καθώς περιέχει το πρωτεύον κλειδί του έργου στο οποίο ανήκει το αρχείο. Είναι προφανές ότι το συγκεκριμένο γνώρισμα χρησιμοποιείται με την λογική ξένου κλειδιού συνδέοντας 1 έργο από τον πίνακα "projects" με n αρχεία από τον πίνακα "project_files". Το γνώρισμα "compiled" είναι τύπου "tinyint(1)" και λειτουργεί ως δείκτης μεταγλώττισης. Με την τιμή 0 φαίνεται ότι δεν έχει γίνει μεταγλώττιση, με την τιμή 1 φαίνεται ότι έχει γίνει μεταγλώττιση και με την τιμή 2 φαίνεται ότι έχει γίνει αλλαγή των περιεχομένων του αρχείου από την τελευταία μεταγλώττιση. Το γνώρισμα "component" είναι τύπου "int(10)" και λειτουργεί ως ξένο κλειδί στον πίνακα βιβλιοθηκών και εξαρτημάτων. Το γνώρισμα αυτό θα έχει την τιμή 0 εάν το αρχείο έχει δημιουργηθεί από τον χρήστη ενώ θα φέρει την τιμή του πρωτεύον κλειδιού της αντίστοιχης εγγραφής από τον πίνακα "libraries" εάν το αρχείο έχει προστεθεί στο έργο από τα διαθέσιμα εξαρτήματα. Ουσιαστικά το γνώρισμα αυτό συνδέει το αρχείο με το εξάρτημα ή βιβλιοθήκη από το οποίο προήρθε. Τέλος, το γνώρισμα "version" είναι τύπου "smallint(4)". Ο τύπος "smallint" συγκρατεί ακαίρους σε 2 Byte επιτρέποντας τιμές από 32,768 ως -32,768. Το γνώρισμα αυτό περιέχει την έκδοση του αρχείου εάν είναι αντίγραφο βιβλιοθήκης ή εξαρτήματος, αλλιώς έχει την τιμή 0. Η συγκράτηση της έκδοσης στον πίνακα του αρχείου, σε συνδυασμό με τον σύνδεσμο για την αντίστοιχη εγγραφή του πίνακα "libraries" μας επιτρέπει την ενημέρωση του χρήστη για την ύπαρξη νέων εκδόσεων στο εξάρτημα που χρησιμοποιεί. Βλέπουμε λοιπόν πως ο πίνακας "project_files" δεν περιέχει μόνο τις απαραίτητες πληροφορίες των αρχείων μας αλλά και στοιχεία που μας επιτρέπουν την ανάπτυξη προχωρημένων λειτουργιών.

3.1.4 Περιγραφή του Πίνακα sid_files

Ο πίνακας "sid_files" αποσκοπεί στην διατήρηση εγγραφών για την διατήρηση αρχείων επισκεπτών. Καθώς οι επισκέπτες έχουν περιορισμένες λειτουργίες, δεν υπάρχει σύνδεση σε κάποιο έργο ή εξάρτημα. Αντίθετα, τα αρχεία αυτά είναι συνδεμένα απευθείας με τον

επισκέπτη μέσω του αναγνωριστικού του αριθμού SID. Έτσι βλέπουμε και στην Εικόνα 7 πως ο πίνακας αυτός αντικατοπτρίζει τον πίνακα "project_files" με έλλειψη πολλών δευτερεύων γνωρισμάτων αλλά και την εισαγωγή του γνωρίσματος "sid", τύπου "int(10)". Το νέο αυτό γνώρισμα είναι που συνδέει το κάθε αρχείο με τον αντίστοιχο επισκέπτη. Παρόλο των περιορισμένων δυνατοτήτων που προσφέρονται στους επισκέπτες, η διατήρηση των αρχείων τους με τρόπο συμβατό με το υπόλοιπο σύστημα είναι αναγκαία. Έτσι αν και φαινομενικά αποκομμένος, ο πίνακας "sid_files" είναι ένας πίνακας που επιτρέπει μία σωστή πρώτη επαφή με την εφαρμογή μας, προσφέροντας αποθηκευτικό χώρο για τις βασικές λειτουργίες που συναντά ένας επισκέπτης.

Name	Type	Collation	Attributes	Null	Default	Extra
id	int(10)			No	*None*	AUTO_INCREMENT
sid	int(10)			No	*None*	
name	varchar(25)	utf8_general_ci		No	*None*	
compiled	tinyint(1)			No	0	

Εικόνα 7: Η δομή του πίνακα sid_files

3.1.5 Περιγραφή του Πίνακα projects_editors

Ο πίνακας "projects_editors" είναι ένας πίνακας διασύνδεσης. Επιτρέπει την σύνδεση χρηστών με τα έργα. Παρόλο που κάτι τέτοιο είναι εφικτό με την χρήση γνωρισμάτων της λογικής ξένου κλειδιού, θα χρειαζόταν σημαντική επεξεργασία των στοιχείων τους και δεν θα επέτρεπε την εισαγωγή βαθμίδων δικαιωμάτων. Για παράδειγμα, είναι δυνατό σε κάθε έργο να έχουμε το γνώρισμα "editor" όπου θα κρατήσουμε το πρωτεύον κλειδί για τον αντίστοιχο χρήστη από τον πίνακα "users". Αυτό όμως δεν μας επιτρέπει την εισαγωγή πολλαπλών συγγραφών. Κάτι τέτοιο θα μπορούσε να γίνει χωρίζοντας τα πρωτεύων κλειδιά τους με ερωτηματικά ";". Αυτή η αντιμετώπιση όμως οδηγεί στην ανάγκη επεξεργασίας των δεδομένων του γνωρίσματος τόσο για την ίδια την ανάγνωση όσο και για την επεξεργασία του.

Name	Type	Collation	Attributes	Null	Default	Extra
id	int(10)			No	*None*	AUTO_INCREMENT
user_id	int(10)			No	*None*	
project_id	int(10)			No	*None*	
user_type	tinyint(1)			No	*None*	

Εικόνα 8: Η δομή του πίνακα projects_editors

Έτσι γίνεται φανερό ότι ένα πίνακας διασύνδεσης μπορεί πολλές φορές να απλοποιήσει απαιτητικές συνδέσεις μεταξύ πινάκων επιτρέποντας παράλληλα την εισαγωγή επιπλέον στοιχείων που περιγράφουν καλύτερα το είδος σύνδεσης των εγγραφών.

Όπως φαίνεται στην Εικόνα 8 ο πίνακας "projects_editors" αποτελεί μέρος του κανόνα χρησιμοποιώντας το γνώρισμα "id" ως πρωτεύον κλειδί. Το δεύτερο γνώρισμα είναι το "user_id", τύπου "int(10)". Το γνώρισμα αυτό συνδέει την εγγραφή με τον πίνακα "users" αντιπροσωπεύοντας το αντίστοιχο πρωτεύον κλειδί της εγγραφής του χρήστη. Το επόμενο γνώρισμα είναι το "project_id", τύπου "int(10)". Με παρόμοιο τρόπο, αυτό το γνώρισμα αντιπροσωπεύει τις εγγραφές του πίνακα "projects". Το γνώρισμα "user_type", τύπου "tinyint(1)" επιτρέπει την εισαγωγή επιπλέων πληροφορίας στην διασύνδεση. Συγκεκριμένα με την τιμή 1 επισημαίνεται ότι η σύνδεση αυτή είναι μεταξύ αρχικού ιδιοκτήτη και έργου, αντίθετα με την τιμή 0 επισημαίνεται ότι η σύνδεση είναι μεταξύ επιπλέον συγγραφέα και έργου. Έτσι το γνώρισμα αυτό μας επιτρέπει την επιβολή διαφορετικών δικαιωμάτων και επιπέδων διασύνδεσης.

3.1.6 Περιγραφή του Πίνακα libraries

Ο πίνακας "libraries" περιέχει γνωρίσματα που περιγράφουν την κάθε βιβλιοθήκη ή εξάρτημα. Παρόμοια με τον πίνακα αρχείων, ο πίνακας "libraries" μας επιτρέπει την συγκράτηση στοιχείων που μας δίνουν την δυνατότητα για πολλές χρήσιμες λειτουργίες. Εάν αυτές οι λειτουργίες δεν υπήρχαν στους στόχους της εφαρμογής, θα μπορούσαμε να χρησιμοποιήσουμε κλήσεις συστήματος για να προσπελάσουμε τα αρχεία βιβλιοθηκών και εξαρτημάτων. Παρακάτω θα εξετάσουμε την δομή του πίνακα "libraries" όπως φαίνεται στην Εικόνα 9.

Name	Type	Collation	Attributes	Null	Default	Extra
id	int(10)			No	*None*	AUTO_INCREMENT
name	varchar(50)	utf8_general_ci		No	*None*	
approved	tinyint(1)			No	0	
owner_id	int(10)			No	*None*	
file_id	int(10)			No	*None*	
version	smallint(4)			No	1	
pending_suggestion	tinyint(1)			No	0	
downloads	int(10)			No	0	

Εικόνα 9: Η δομή του πίνακα libraries

Αρχικά βλέπουμε ότι το πρώτο γνώρισμα είναι ως συνήθως το "id" που αποτελεί το αναγνωριστικό γνώρισμα και πρωτεύων κλειδί του πίνακα. Το δεύτερο γνώρισμα είναι το

"name", τύπου "varchar(50)" που περιέχει το όνομα και κατάληξη κάθε αρχείου βιβλιοθήκης. Έπειτα έχουμε το γνώρισμα "approved", τύπου "tinyint(1)". Το συγκεκριμένο γνώρισμα παίρνει την τιμή 0 όταν η βιβλιοθήκη εκκρεμεί την έγκριση ενός διαχειριστή και την τιμή 1 όταν έχει εγκριθεί. Οι χρήστες μπορούν να βρουν και να εισάγουν στα έργα τους μόνο βιβλιοθήκες που έχουν εγκριθεί. Το επόμενο γνώρισμα είναι το "owner_id", τύπου "int(10)". Το γνώρισμα αυτό λειτουργεί ως ξένο κλειδί στον πίνακα "users" όπου συνδέεται με το "id" της αντίστοιχης εγγραφής. Η σύνδεση είναι τύπου 1 χρήστη με n βιβλιοθήκες. Το νόημα της σύνδεσης είναι να μας επιτρέψει να συγκρατήσουμε τον χρήστη που αρχικά δημιούργησε και δημοσίευσε την βιβλιοθήκη. Παρόμοια το γνώρισμα "file_id", τύπου "int(10)" συνδέει αυτόν τον πίνακα με τον "project_files" περιέχοντας το "id" του αρχείου από το οποίο προήρθε η βιβλιοθήκη. Το γνώρισμα "version" είναι τύπου "smallint(4)" και συγκρατεί την έκδοση της βιβλιοθήκης. Το γνώρισμα "pending_suggestion", τύπου "tinyint(1)" παίρνει τις τιμές 0 και 1. Η τιμή 0 σημαίνει ότι δεν υπάρχει κάποια πρόταση για αλλαγές ενώ αντίθετα, η τιμή 1 σημαίνει ότι υπάρχει πρόταση για αλλαγές. Τέλος, το γνώρισμα "downloads" με τύπο "int(10)" συγκρατεί τον αριθμό που οι χρήστες έχουν εισάγει την βιβλιοθήκη σε κάποιο έργο τους. Λειτουργώντας με την λογική ότι οι χρήσιμες βιβλιοθήκες εισάγονται περισσότερες φορές, μπορούμε να χρησιμοποιήσουμε αυτό το γνώρισμα για την ταξινόμηση αποτελεσμάτων αναζήτησης. Έτσι, με τον πίνακα "libraries" μπορούμε να περιγράψουμε ικανοποιητικά την κάθε βιβλιοθήκη. Με το συγκεκριμένο σύνολο γνωρισμάτων αποσκοπούμε στην συγκράτηση δεδομένων τόσο για απλές όσο και για πιο προηγμένες λειτουργίες που δημιουργούν ένα προηγμένο σύστημα βιβλιοθηκών.

3.1.7 Περιγραφή του Πίνακα library_updates

Οι βιβλιοθήκες δεν είναι στατικές αλλά αναβαθμίζονται και αναπτύσσονται συνεχώς. Η διόρθωση πιθανών προβλημάτων και περαιτέρω ανάπτυξη γίνεται συνήθως από τον αρχικό συγγραφέα τόσο για θέματα πνευματικής ιδιοκτησίας όσο και για θέματα οικειότητας με τον κώδικα και τον σκοπό της βιβλιοθήκης. Έτσι χρησιμοποιούμε τον πίνακα "library_updates" για την διευκόλυνση αυτής της λειτουργίας. Ο πίνακας αυτός έχει ως σκοπό την διευκόλυνση διατήρησης προσωρινών αντιγράφων. Τα προσωρινά αυτά αντίγραφα είναι οι προτάσεις για αναβάθμιση από τους ιδιοκτήτες των βιβλιοθηκών. Για την ύπαρξη όμως μίας ομαλής λειτουργίας του συστήματος βιβλιοθηκών, τέτοιες προτάσεις ελέγχονται και εγκρίνονται ή απορρίπτονται από τους διαχειριστές. Έτσι η ύπαρξη προσωρινών αρχείων μέχρι να ολοκληρωθεί η διαδικασία είναι απαραίτητη.

Name	Type	Collation	Attributes	Null	Default	Extra
id	int(10)			No	*None*	AUTO_INCREMENT
lib_id	int(10)			No	*None*	
library	varchar(50)	utf8_general_ci		No	*None*	

Εικόνα 10: Η δομή του πίνακα library_updates

Ο πίνακας “library_updates” αποτελείται από τρία γνωρίσματα όπως φαίνεται στην Εικόνα 10. Αρχικά έχουμε το πρωτεύον κλειδί, “id” που χρησιμοποιείται ως συνήθως για την ταυτοποίηση κάθε εγγραφής στον πίνακα. Έπειτα έχουμε το γνώρισμα “lib_id”, τύπου “int(10)” που λειτουργεί ως ξένο κλειδί συνδέοντας με τον πίνακα “libraries” στο αντίστοιχο “id”. Έτσι δημιουργούμε την σύνδεση μεταξύ προσωρινού αρχείου και βιβλιοθήκης στην οποία αντιστοιχεί. Τέλος έχουμε το γνώρισμα “library”, τύπου “varchar(50)” που περιέχει το όνομα του προσωρινού αρχείου. Έτσι, με έναν απλό έλεγχο του πίνακα αυτού μπορούμε να βρούμε άμεσα τις προτεινόμενες αλλαγές για κάθε βιβλιοθήκη καθώς και το αρχείο που τις περιέχει.

Ολοκληρώνοντας την περιγραφή της βάσης δεδομένων, αποκτούμε μια αρχική εικόνα του τρόπου οργάνωσης της εφαρμογής μας. Παρόλα αυτά, υπάρχουν δεδομένα που δεν ανήκουν στην βάση δεδομένων αλλά η συγκράτησή τους είναι ζωτικής σημασίας για τις λειτουργίες που χρησιμοποιούμε. Τέτοια δεδομένα παίρνουν την μορφή αρχείων και όπως θα δούμε στο επόμενο υποκεφάλαιο, είναι οργανωμένα σε δομές καταλόγων.

3.2 Κατάλογοι Αρχείων

Η εφαρμογή που αναπτύξαμε έχει ως βασικό στόχο την δημιουργία, μεταγλώττιση και προσομοίωση προγραμμάτων. Αυτές οι διαδικασίες αποτελούνται από ενέργειες που εφαρμόζονται σε αρχεία. Από την δημιουργία και επεξεργασία των αρχείων VHDL έως την εξαγωγή μεταγλωττισμένων αρχείων και αρχείων αποτελεσμάτων προσομοίωσης. Όπως είδαμε παραπάνω, η σωστή δομή δεδομένων είναι ένας σημαντικός καταλύτης για την σωστή λειτουργία της εφαρμογής. Η λογική αυτή ισχύει και στην δομή καταλόγων. Παρακάτω θα εξηγήσουμε την δομή των καταλόγων που χρησιμοποιούμε. Έχουμε χωρίσει αρχικά τους καταλόγους σε μόνιμους και προσωρινούς. Έπειτα διακρίνεται η κατηγορία χρήσης κάθε καταλόγου. Αρχικά θα επεκτείνουμε την δομή των μόνιμων καταλόγων με τα αρχεία χρηστών και βιβλιοθηκών. Μετά θα δούμε τον τρόπο διατήρησης των προσωρινών καταλόγων κατηγοριοποιημένων σε αρχεία επισκεπτών, αρχεία εργασιών προσομοίωσης και μεταγλώττισης και τέλος σε αρχεία αναφοράς κατάστασης.

3.2.1 Αρχεία Χρηστών και Βιβλιοθηκών

Οι μόνιμοι κατάλογοι που χρησιμοποιούμε διακρίνονται ως προς τη χρήση για την διατήρηση αρχείων χρηστών και ως προς τη χρήση για την διατήρηση αρχείων βιβλιοθηκών. Όπως θα δούμε παρακάτω, οι κατάλογοι αυτοί βρίσκονται στο ίδιο επίπεδο του βασικού μόνιμου καταλόγου που βρίσκεται στην διαδρομή “/home/user/hdl-compiler/”.

Ο κατάλογος χρηστών περιέχει τα αρχεία κάθε χρήστη. Κατά την δημιουργία ενός λογαριασμού χρήστη, δημιουργείται στην παραπάνω διαδρομή ένας νέος κατάλογος με ονομασία που λαμβάνεται από το όνομα του χρήστη. Έτσι, εάν ο χρήστης έχει δώσει ως όνομα “User” θα δημιουργηθεί ο κατάλογος “/home/user/hdl-compiler/User/”. Όταν ο χρήστης δημιουργήσει ένα έργο, τότε χρησιμοποιώντας την ονομασία του έργου, κατάλληλα τροποποιημένη, θα δημιουργηθεί και κατάλογος στον οποίο θα διατηρούνται τα αρχεία αυτού του έργου. Εάν για παράδειγμα ο χρήστης “User” δημιουργήσει το έργο “Project 1”, θα

δημιουργηθεί και ο κατάλογος "/home/user/hdl-compiler/User/Project_1/". Σε αυτόν τον κατάλογο θα προστίθενται τα αρχεία που δημιουργεί ή εισάγει στο έργο "Project 1" ο χρήστης. Επίσης σε αυτόν τον κατάλογο θα δημιουργηθούν και τα αρχεία μεταγλώττισης και προσομοίωσης για το ίδιο έργο. Έτσι χρησιμοποιούμε έναν συγκεκριμένο κατάλογο για την οργάνωση των αρχείων χρηστών μέσα στον οποίο αναπτύσσεται η δομή καταλόγων για τον κάθε χρήστη και το κάθε έργο που χρησιμοποιεί.

Μέσα στον βασικό μόνιμο κατάλογο, διατηρούμε και δύο ακόμη καταλόγους για την χρήση βιβλιοθηκών. Αρχικά έχουμε τον κατάλογο "/home/user/hdl-compiler/libraries/" στον οποίο βρίσκονται τα αρχεία βιβλιοθηκών. Κατά την δημοσίευση μίας βιβλιοθήκης, το αρχείο αντιγράφεται σε αυτόν τον κατάλογο και αναμένει έγκριση ή απόρριψη από έναν διαχειριστή. Κατά την εισαγωγή ή ενημέρωση μίας βιβλιοθήκης σε ένα έργο, έχουμε την αντιγραφή του αρχείου στον φάκελο του έργου. Ο δεύτερος κατάλογος που χρησιμοποιούμε για την λειτουργία βιβλιοθηκών είναι ο "/home/user/hdl-compiler/update_libraries/". Όταν ο συγγραφέας μίας βιβλιοθήκης προτείνει μία τροποποίηση, το αντίστοιχο αρχείο από το προσωπικό του έργο αντιγράφεται σε αυτόν τον κατάλογο και αναμένει έγκριση ή απόρριψη. Εάν γίνει έγκριση της τροποποίησης από έναν διαχειριστή, το αρχείο θα μεταφερθεί και αντικαταστήσει το αντίστοιχο στον κατάλογο βιβλιοθηκών. Στην περίπτωση απόρριψης, το αρχείο θα διαγραφεί από τον κατάλογο προτάσεων. Η δομή αυτή μας επιτρέπει την χρήση του υπάρχοντος μόνιμου καταλόγου για την διατήρηση των απαραίτητων αρχείων για τις λειτουργίες των βιβλιοθηκών. Παράλληλα προσθέτει τον περιορισμό κατάληψης των ονομάτων "libraries" "update_libraries" από τα πιθανά ονόματα χρηστών. Θεωρούμε όμως ότι ο περιορισμός αυτός είναι αμελητέος δεδομένου του αριθμού διαθέσιμων ονομάτων κι έτσι γίνεται φανερό ότι το πλεονέκτημα χρήσης υπάρχοντος καταλόγου υπερτερεί.

3.2.2 Αρχεία Επισκεπτών

Τα αρχεία επισκεπτών έχουν παρόμοια δομή με τα αρχεία χρηστών. Η βασική διαφορά που παρατηρείται είναι ότι στα αρχεία επισκεπτών η έννοια του χρήστη και έργο συμπίπτει. Συγκεκριμένα, τα αρχεία επισκεπτών διατηρούνται στον κατάλογο "/tmp/VHDL/". Όταν ένας επισκέπτης έρχεται σε πρώτη επαφή με την εφαρμογή μας, του παρέχεται ένας τυχαίος αλλά μοναδικός αριθμός συνοδού (session ID). Ταυτόχρονα δημιουργείται ένας κατάλογος χρησιμοποιώντας τον ίδιο αριθμό ως "/tmp/VHDL/SID/". Οι επισκέπτες δεν μπορούν να δημιουργήσουν επιπλέον έργα κι έτσι χρησιμοποιείται αυτός ο κατάλογος απευθείας για την διατήρηση των αρχείων τους. Σε μελλοντικές επαφές με την εφαρμογή μας, ο επισκέπτης, εάν δεν έχει δημιουργήσει δικό του λογαριασμό ακόμα, μπορεί να συνδεθεί στον προσωρινό "SID" που του δόθηκε μέχρι την επόμενη επανεκκίνηση του εξυπηρετητή. Τότε τα αρχεία που βρίσκονται σε προσωρινούς καταλόγους θα απομακρυνθούν. Το σύστημα αυτό επιτρέπει στους επισκέπτες να δοκιμάσουν κάποιες σημαντικές λειτουργίες της εφαρμογής μας αλλά με τον συμβιβασμό περιορισμένων και προσωρινών πόρων.

3.2.3 Αρχεία Εργασιών και Αναφοράς Κατάστασης

Τόσο οι χρήστες όσο και οι επισκέπτες έχουν πρόσβαση στις λειτουργίες μεταγλώττισης και προσομοίωσης. Οι λειτουργίες αυτές κάνουν χρήση του προσωρινού καταλόγου "/tmp/jobs/". Όταν ένας χρήστης ή επισκέπτης κάνει αίτηση για προσομοίωση ή μεταγλώττιση ενός αρχείου, δημιουργείται ένα αρχείο εργασίας σ' αυτόν τον προσωρινό κατάλογο. Καθώς το σενάριο τερματικού "job_scheduler.sh" εκτελείται, ελέγχει τον κατάλογο "/tmp/jobs/" για αρχεία εργασίας. Όταν ένα αρχείο εργασίας βρεθεί, εκτελείται η λειτουργία του και το αρχείο αυτό απομακρύνεται. Έτσι χρησιμοποιούμε ουσιαστικά τον κατάλογο "/tmp/jobs/" ως ουρά αναμονής για τις εργασίες μεταγλώττισης και προσομοίωσης. Για την παρακολούθηση τρέχουσας εργασίας αλλά και της ουράς αναμονής χρησιμοποιούμε το αρχείο αναφοράς κατάστασης. Το αρχείο αυτό δημιουργείται από το σενάριο τερματικού στον κατάλογο "/tmp/status/" και περιλαμβάνει δεδομένα σχετικά με την κατάσταση εργασίας. Σε περίπτωση που δεν εκτελείται κάποια εργασία υπάρχει αντίστοιχο μήνυμα αλλιώς μας δίνει το αναγνωριστικό "ID" του αρχείου εργασίας που εκτελείται και τον αριθμό υπολειπόμενων αρχείων εργασίας στην ουρά αναμονής. Η διατήρηση καταλόγων και αρχείων γι' αυτές τις λειτουργίες μας επιτρέπει την επαφή του σεναρίου τερματικού και των εργαλείων που χρησιμοποιεί με την υπόλοιπη εφαρμογή, ανεπτυγμένη σε τεχνολογίες διαδικτύου, γεφυρώνοντας τα προβλήματα συμβατότητας μεταξύ τους.

Κλείνοντας την δομή των καταλόγων, πιστεύουμε ότι δείξαμε τον τρόπο χρήσης του δεύτερου μέρους των αποθηκεμένων αρχείων. Σε συνδυασμό με την βάση δεδομένων, έχουμε μία πλήρη εικόνα των στοιχείων που βρίσκονται στον εξυπηρετητή. Στα επόμενα υποκεφάλαια θα δούμε πώς τα στοιχεία αυτά αξιοποιούνται για την υποστήριξη των λειτουργιών που συνθέτουν την εφαρμογή μας.

3.3 Ανάπτυξη και Υλοποίηση της Εφαρμογής

Σε προηγούμενα κεφάλαια αναφέραμε τα εργαλεία που χρησιμοποιήθηκαν για την ανάπτυξη της εφαρμογής. Παρακάτω θα εξηγήσουμε τον τρόπο που χρησιμοποιήσαμε αυτά τα εργαλεία για την σύνταξη των λειτουργιών που συνθέτουν την εφαρμογή μας. Οι λειτουργίες παρουσιάζονται κατηγοριοποιημένες με τον σκοπό που επιτελούν αλλά και με τον τρόπο υλοποίησής τους. Αρχικά κάνουμε την ανάλυση των λειτουργιών που είναι απαραίτητες για την ορθή φόρτωση της εφαρμογής και την μετάφραση του συνδέσμου στην σελίδα που ζητήθηκε. Έπειτα εξετάζουμε τον τρόπο υλοποίησης των λειτουργιών που αναλαμβάνουν την λειτουργική αλληλεπίδραση του χρήστη με τον εξυπηρετητή. Οι επόμενες λειτουργίες που βλέπουμε αποτελούνται από κλάσεις και μεθόδους. Προχωράμε μετά στην ανάλυση του τρόπου υλοποίησης και χρήσης του θέματος εμφάνισης και τέλος εξηγούμε τις τεχνικές που χρησιμοποιήσαμε για την κάλυψη της εφαρμογής από θέματα ασφάλειας.

3.3.1 Αρχικοποίηση της Εφαρμογής κατά την Φόρτωση

Όπως είδαμε στο θεωρητικό υπόβαθρο η PHP είναι μία γλώσσα σεναρίου. Έτσι, κάθε φορά που ο χρήστης φορτώνει μία σελίδα, ο εξυπηρετητής θα κάνει μία καινούρια εκτέλεση του σεναρίου ώστε να προσφέρει το αντίστοιχο αποτέλεσμα. Χρησιμοποιούμε λοιπόν στην εφαρμογή μας δύο αρχεία που έχουν ως στόχο την ομαλή λειτουργία κάθε σεναρίου και την δημιουργία των απαραίτητων συνδέσεων μεταξύ διαφορετικών εκτελέσεων.

Το πρώτο αρχείο που εξετάζουμε είναι το loader.php. Το συγκεκριμένο αρχείο είναι υπεύθυνο για την σωστή αρχικοποίηση του σεναρίου.

```
1. // Initiate session
2. if (session_status() == PHP_SESSION_NONE) {
3. session_start();
4. }
```

Κώδικας 1: loader.php - Έλεγχος και δημιουργία συνεδρίας

Αρχικά, ελέγχουμε εάν υπάρχει ήδη η συνεδρία αλλιώς την δημιουργούμε (Κώδικας 1) ώστε να έχουμε διαθέσιμες τις μεταβλητές που πιθανός να περιλαμβάνει.

```
1. # Set here the variables
2. # --START--
3. $BASE="/tmp/VHDL/";
4. $JOBDIRECTORY="/tmp/jobs/";
5. $STATUSDIR="/tmp/status/";
6. $protocol = (!empty($_SERVER['HTTPS']) && $_SERVER['HTTPS'] !== 'off'
   || $_SERVER['SERVER_PORT'] == 443) ? "https://" : "http://";
7. $BASE_URL = $protocol.$_SERVER['SERVER_NAME']."/vhdl";
8. $BASE_DIR = "/home/user/hdl-compiler/";
9. $BASE_SID = "/tmp/VHDL/";
10. # --END--
11.
12. //Check the existance of the BASE directory and the JOBDIRECTORY
13. if ( ! file_exists($BASE) )
14.   mkdir($BASE,0777,true);
15. if ( ! file_exists($JOBDIRECTORY) )
16.   mkdir($JOBDIRECTORY,0777,true);
17. if ( ! file_exists($STATUSDIR) )
18.   mkdir($STATUSDIR,0777,true);
```

Κώδικας 2: loader.php - Αρχικοποίηση καταλόγων

Έπειτα, καθορίζουμε βασικές καθολικές μεταβλητές που περιέχουν τους καταλόγους που χρησιμοποιούμε αλλά και τον διαδικτυακό σύνδεσμο της ιστοσελίδας (Κώδικας 2, γραμμές 1-9). Αφού θέσουμε αυτές τις μεταβλητές, προχωράμε στον έλεγχο για την ύπαρξη των προσωρινών καταλόγων και την δημιουργία τους εάν είναι απαραίτητο (Κώδικας 2, γραμμές 13-18).

```
1. // Load Classes
2. require_once('classes/Database.php');
3. require_once('classes/Messages.php');
4. require_once('classes/User.php');
5. require_once('classes/General.php');
6. // init. objects
7. $db = new Database;
8. $messages = new Messages;
9. $gen = new General;
```

Κώδικας 3: loader.php - Αρχικοποίηση αντικειμένων

Το επόμενο βήμα είναι η αρχικοποίηση των αντικειμένων που χρησιμοποιούμε. Αρχικά καλούμε τα αρχεία που περιέχουν τις αντίστοιχες κλάσεις (Κώδικας 3, γραμμές 2-5) και αμέσως μετά δημιουργούμε τα αντικείμενα που θα χρησιμοποιήσουμε.

```
1. // login pseudo user with session id
2. if(isset($_SESSION['SID']) && $_SESSION['SID']>0){
3.   $_SESSION['vhdl_user']['username'] = "Guest";
4.   $_SESSION['vhdl_user']['id'] = "0";
5.   $_SESSION['vhdl_user']['loged_in'] = 1;
6.   $_SESSION['vhdl_user']['type'] = 0;
7. }
8. if( isset($_SESSION['vhdl_user']) ){
9.   $user = new User($_SESSION['vhdl_user']);
10. }
11. if( !isset($_SESSION['vhdl_msg']) ){
12.   $_SESSION['vhdl_msg'] = array();
13. }
```

Κώδικας 4: loader.php - Αρχικοποίηση συνεδρίας και αντικειμένου χρήστη

Συγκεκριμένα για την δημιουργία του αντικείμενου χρήστη πρέπει να προηγηθεί ο έλεγχος επισκέπτη. Εάν έχουμε σύνδεση επισκέπτη θα θέσουμε τις κατάλληλες τιμές συνεδρίας (Κώδικας 4, γραμμές 2-7). Με παρόμοιο τρόπο θέτουμε τις τιμές συνεδρίας κατά την σύνδεση ενός χρήστη. Έτσι, μπορούμε μετά να ελέγξουμε εάν υπάρχει συνδεμένος χρήστης ή επισκέπτης και να δημιουργήσουμε το αντικείμενο χρήστη (Κώδικας 4, γραμμές 8-10). Αμέσως μετά έχουμε την αρχικοποίηση της συνεδρίας μηνυμάτων σε περίπτωση που δεν υπάρχει ήδη (Κώδικας 4, γραμμές 11-13). Η συγκεκριμένη συνεδρία μεταφέρει μηνύματα ανατροφοδότησης που ενημερώνουν τον χρήστη για το αποτέλεσμα της λειτουργίας που επιχείρησε.

```
1. require_once('security.php');
2. require_once('post_handler.php');

3. ?>
```

Κώδικας 5: loader.php - Κλήση αρχείων ασφαλείας και αλληλεπίδρασης

Τέλος έχουμε την κλήση του αρχείου security.php, σημαντικό μέρος της ασφάλειας στην εφαρμογή μας και του αρχείου post_handler.php, υπεύθυνο για την διαχείριση αλληλεπίδρασης με τον χρήστη στην αποστολή φόρμας (Κώδικας 5). Βλέπουμε λοιπόν ότι το αρχείο loader.php αποτελεί την αρχικοποίηση βασικών στοιχείων όπως οι συνεδρίες, καθολικές μεταβλητές και κλάσεις που απαιτούνται σε κάθε εκτέλεση ενός σεναρίου της εφαρμογής μας.

Το δεύτερο αρχείο που θα αναλύσουμε είναι το index.php. Το συγκεκριμένο αρχείο αποτελεί κομβικό σημείο της εφαρμογής μας και δρομολογεί την αίτηση κίνησης στην κατάλληλη σελίδα. Η δρομολόγηση της κίνησης σε μία ιστοσελίδα μπορεί να επιτευχθεί με πολλούς τρόπους. Ο συνηθέστερος τρόπος είναι απευθείας, μέσω της διεύθυνσης διαδικτύου όπου στοχεύει σε ξεχωριστά αρχεία. Με αυτόν τον τρόπο ο χρήστης μπορεί εύκολα να προσπελάσει απευθείας μία συγκεκριμένη σελίδα αφού θα έχει μοναδική διαδικτυακή διεύθυνση στοχεύοντας σε συγκεκριμένο αρχείο. Ταυτόχρονα όμως κάθε αρχείο της σελίδας πρέπει να περιέχει ένα ολοκληρωμένο στιγμιότυπο της εφαρμογής. Ένας άλλος τρόπος είναι άμεση λήψη της ζητούμενης σελίδας με την τεχνική AJAX και την εμφάνισή της στο υπάρχον παράθυρο, συνήθως μέσα σε υπάρχον στοιχείο HTML, χωρίς την ανάγκη επαναφόρτισης. Με αυτόν τον τρόπο η μεταβίβαση από την μία σελίδα στην άλλη είναι άμεση και μεταφορτώνονται μόνο τα στοιχεία που αλλάζουν. Έτσι όμως χρειάζεται και η περαιτέρω ανάπτυξη ενός συστήματος δρομολόγησης σε JavaScript για να επιτραπεί στον χρήστη η απευθείας μετάβαση σε συγκεκριμένες σελίδες. Ο τρόπος δρομολόγησης της κίνησης που χρησιμοποιούμε στην εφαρμογή μας στοχεύει σε έναν επιλεκτικό συνδυασμό των δύο παραπάνω. Χρησιμοποιούμε την λογική του δεύτερου τρόπου, δημιουργώντας ένα κομβικό σημείο όπου δρομολογεί την κίνηση. Οι αιτήσεις δρομολόγησης γίνονται μέσω της διαδικτυακής διεύθυνσης όπου περιέχει την ζητούμενη σελίδα ως μεταβλητή αιτήματος $_GET επιτρέποντας στον χρήστη να μεταβεί άμεσα σε οποιαδήποτε σελίδα.

```
1. <?php
2. // load necessary files
3. include('loader.php');
4. 
5. // define actions allowed to any visitor
6. $allowed_actions = array("project-file");
7. if(isset( $_GET['action'] )){
8.   $allowed_for_visitor = in_array($_GET['action'], $allowed_actions);
9. }else{
10.   $allowed_for_visitor = false;
11. }
12. 
13. // display the appropriate (or requested) page for logged in and out
    users
14. if( (isset($_SESSION['vhdl_user']['loged_in']) && $_SESSION['vhdl_us-
    er']['loged_in']==1) || $allowed_for_visitor ){
15.   if( isset($_GET['action']) ){
16.     $included_file = $_GET['action'].'.php';
17.   }else{
18.     if($_SESSION['vhdl_user']['id'] == '0'){
19.       $included_file='loggedin_sid.php';
20.     }else{
21.       $included_file='loggedin.php';
22.     }
23.   }
24. }else{
25.   $included_file='loggedout.php';
26. }
```

Κώδικας 6: index.php - Κλήση του loader.php και δρομολόγηση

Ταυτόχρονα όμως μέσω του κόμβου θέτουμε ελέγχους και φίλτρα επιβεβαιώνοντας ότι η σελίδα υπάρχει και είναι διαθέσιμη για τον χρήστη. Έτσι μπορούμε να καλέσουμε συγκεκριμένα αρχεία κατά την ροή της κίνησης μέσω των κόμβων που θα εκτελέσουν απαραίτητες λειτουργίες σε κάθε επαφή με τον χρήστη και να ξεχωρίσουμε αυτές τις λειτουργίες από τα τελικά αρχεία στα οποία βρίσκονται οι διαφορετικές υπηρεσίες κάθε σελίδας. Έτσι, αρχικά έχουμε την κλήση του αρχείου loader.php (Κώδικας 6, γραμμή 3) και μετά προχωράμε στην δρομολόγηση. Μερικές σελίδες επιτρέπουν την πρόσβαση από κάθε επισκέπτη, είτε είναι συνδεμένος ή όχι, ορίζουμε αυτές τις σελίδες σε έναν πίνακα και έπειτα συγκρίνουμε την σελίδα που ζητήθηκε με αυτές στον πίνακα (Κώδικας 6, γραμμές 6-11). Η σύγκριση μας επιστρέφει τιμές ψευδής ή αληθής. Στην συνέχεια ελέγχουμε εάν ο χρήστης είναι συνδεμένος ή ο επισκέπτης επιτρέπεται να προσπελάσει την συγκεκριμένη σελίδα και εάν έχει ζητηθεί σελίδα, την επιλέγουμε, αλλιώς προσφέρουμε την αντίστοιχη αρχική σελίδα για το είδος του χρήστη ή επισκέπτη (Κώδικας 6, γραμμές 14-26).

```
1. //bring header and navbar
2. include('theme/header.php');
3. // display messages from the user's last interaction
4. $messages->display_msg($_SESSION['vhdl_lang'],$_SESSION['vhdl_msg']);
5. //bring the requested file
6. include('pages/'.$included_file);
7. //bring the footer
8. include('theme/footer.php');
9.
10. ?>
```

Κώδικας 7: index.php -Κλήση αρχείων θέματος εμφάνισης και εμφάνιση μηνυμάτων

Στην συνέχεια καλούμε τα αρχεία θέματος εμφάνισης header.php και footer.php τα οποία περικλείουν το αρχείο που επιστρέφει τις λειτουργίες της σελίδας που ζητήθηκε και εάν υπάρχει μήνυμα ανάδρασης, τοποθετείται κατάλληλα στην σελίδα (Κώδικας 7). Ολοκληρώνεται λοιπόν με αυτόν τον τρόπο η δρομολόγηση προσφέροντας το αντίστοιχο αρχείο, μέσα στο θέμα εμφάνισης που χρησιμοποιούμε, αφού έχει ελεγχθεί η κίνηση και έχουν αρχικοποιηθεί οι κατάλληλες λειτουργίες.

Έτσι ολοκληρώνονται οι πρώτες διαδικασίες της εφαρμογής σε κάθε φόρτωση. Έχουμε λοιπόν το αρχείο loader.php, υπεύθυνο για την εισαγωγή των απαραίτητων λειτουργιών και το αρχείο index.php όπου δρομολογούμε την κίνηση με συγκεκριμένη και ελεγχόμενη ροή. Βλέποντας αυτά τα αρχεία, γίνεται μία εισαγωγή τόσο στον τρόπο λειτουργίας όσο και στην λογική σχεδίασης και ανάπτυξης που χρησιμοποιούμε. Στα επόμενα υποκεφάλαια θα δούμε ποιο αναλυτικά κάποια βήματα που συναντάμε ακολουθώντας την ροή κίνησης και τις λειτουργίες που προσφέρουν.

3.3.2 Ανάλυση των Διαχειριστών Αλληλεπίδρασης

Η αλληλεπίδραση του χρήστη με την εφαρμογή γίνεται είτε με την άμεση πλοήγηση σε μία σελίδα είτε με την αποστολή μίας φόρμας και αίτηση πληροφοριών μέσω AJAX. Στο προηγούμενο υποκεφάλαιο περιγράψαμε τον τρόπο που αντιμετωπίζουμε την άμεση

πλοήγηση. Παρακάτω εξηγούμε τις λειτουργίες που αναλαμβάνουν την έμμεση ή άμεση αποστολή φόρμας και την διεπαφή για τις λειτουργίες AJAX.

Αρχικά εξετάζουμε το αρχείο post_handler.php. Το αρχείο αυτό περιέχει τις λειτουργίες που αντιστοιχούν στην διαχείριση αποστολής μίας φόρμας. Η κάθε φόρμα που χρησιμοποιούμε περιέχει ένα HTML στοιχείο με την ονομασία "post_action" και τιμή αντίστοιχη με την λειτουργία που εκτελεί. Έτσι όπως φαίνεται (Κώδικας 8, γραμμή 3) ο πρώτος έλεγχος που γίνεται επιβεβαιώνει ότι υπάρχει HTTP αίτηση POST και ότι έχει τεθεί μία τιμή στον πίνακα "$_POST['post_action']". Αμέσως μετά έχουμε τις λειτουργίες για συνδεμένους χρήστες. Επιβεβαιώνουμε λοιπόν ότι ο χρήστης είναι συνδεμένος (Κώδικας 8, γραμμή 4) και στην συνέχεια χρησιμοποιούμε μία δομή "switch" όπου ελέγχουμε το είδος της λειτουργίας που αιτήθηκε (Κώδικας 8, γραμμές 5-10). Μία δομή "if-else if" θα ήταν εξίσου αποτελεσματική αλλά στην συγκεκριμένη περίπτωση η δομή που χρησιμοποιούμε προσφέρει έναν ποιο ευανάγνωστο κώδικα που διευκολύνει στην συντήρηση και επέκταση.

```
1. <?php
2. //handles the post requests
3. if($_SERVER['REQUEST_METHOD']=='POST' && isset($_POST["post_action"]) ){
4.    if( isset($_SESSION['vhdl_user']['loged_in']) && $_SESSION['vhdl_us-
   er']['loged_in']==1){
5.      switch($_POST["post_action"]){
6.        case "logout":
7.          [...]
8.        break;
9.        [...]
10.     }
11.   }
12.   switch($_POST["post_action"]){
13.     case "login":
14.       [...]
15.     break;
16.     [...]
17.   }
18. }
19. ?>
```

Κώδικας 8: post_handler.php - Γενική δομή του αρχείου

Με παρόμοια δομή ελέγχουμε και τις λειτουργίες που είναι διαθέσιμες σε όλους τους επισκέπτες, όπως η λειτουργία εισόδου σε λογαριασμό (Κώδικας 8, γραμμές 12-17). Έτσι διαχωρίζουμε τις διαθέσιμες λειτουργίες με κατηγοριοποίηση έχοντας ως αποτέλεσμα έναν εύκολα συντηρήσιμο κώδικα που απλοποιεί την διαδικασία προσθήκης ή επεξεργασίας λειτουργιών.

Το δεύτερο αρχείο υπεύθυνο για την διαχείριση αλληλεπίδρασης είναι το "ajax_handler.php". Το αρχείο αυτό λειτουργεί με την ίδια λογική που παρουσιάσαμε στο "post_handler.php" με λίγες διαφορές που εξηγούμε παρακάτω. Η λειτουργία του αρχείου είναι η διαχείριση των αιτήσεων μέσω AJAX. Τα αιτήματα αυτά παρακάμπτουν την κανονική ροή της κίνησης στην εφαρμογή μας και έτσι αποφεύγεται η φόρτωση και εκτέλεση διάφορων σεναρίων που δεν χρησιμοποιούνται όπως το θέμα εμφάνισης.

```
1. <?php
2. // load necessary files
3. include('loader.php');
4.
5. //handles the post requests
6. if($_SERVER['REQUEST_METHOD']=='POST' && isset($_POST["ajax_action"]) &&
   isset($_SESSION['vhdl_user']['loged_in']) && $_SESSION['vhdl_us-
   er']['loged_in']==1){
7.   switch($_POST["ajax_action"]){
8.     case "select_users_like":
9.        [...]
10.    break;
11.    [...]
12.  }
13. }else{
14.   header("Location:".$BASE_URL);
15. }
16. ?>
```

Κώδικας 9: ajax_handler.php - Γενική δομή του αρχείου

Έτσι, το αρχείο "ajax_handler.php" έχει ως πρώτη εντολή την κλήση του αρχείου "loader.php" (Κώδικας 9, γραμμή 3). Καλώντας το αρχείο αρχικοποίησης, έχουμε πρόσβαση στις λειτουργίες, μεταβλητές και συνεδρίες που θα χρειαστούμε. Οι αιτήσεις AJAX που χρησιμοποιούμε γίνονται σε HTTP με την μέθοδο POST και περιέχουν την μεταβλητή "ajax_action" που λειτουργεί με παρόμοιο τρόπο με την μεταβλητή "post_action" του αρχείου "post_handler.php". Στην συνέχεια, γίνεται έλεγχος για την ύπαρξη αίτησης POST και συγκεκριμένα για την ύπαρξη τιμής στον πίνακα "$_POST['ajax_action']" (Κώδικας 9, γραμμή 6). Στον ίδιο έλεγχο επιβεβαιώνουμε ότι υπάρχει συνδεμένος χρήστης καθώς όλες οι διαθέσιμες λειτουργίες AJAX προσφέρονται μόνο σε συνδεμένους λογαριασμούς. Στην περίπτωση που ο χρήστης δεν είναι συνδεμένος ή δεν υπάρχει τιμή στον πίνακα "$_POST['ajax_action']", έχουμε ανακατεύθυνση στην αρχική σελίδα (Κώδικας 9, γραμμές 13-15). Η επιλογή της λειτουργίας γίνεται με την δομή "switch" για τους ίδιους λόγους που εξηγήσαμε στο αρχείο "post_handler.php". Έτσι δημιουργούμε ένα προσαρμοσμένο σύστημα για την αντιμετώπιση αιτήσεων μέσω AJAX που συμβαδίζει στην λογική εύκολα συντηρήσιμου κώδικα.

Τα δύο αυτά αρχεία μας επιτρέπουν την ανάλυση αιτήσεων του χρήστη και επιλογή της σωστής λειτουργίας. Τα αρχεία αυτά περιέχουν την δομή κάθε αλληλεπίδρασης είτε πρόκειται για την δημιουργία λογαριασμού χρήστη ή για την αναζήτηση ενός έργου. Παρόλα αυτά, οι λειτουργίες που αναπτύχθηκαν σε αυτά τα αρχεία περιέχουν βασικότερα δομικά υλικά. Στο επόμενο υποκεφάλαιο βλέπουμε τις κλάσεις που συνθέτουν αυτά τα υλικά στο χαμηλότερο επίπεδο.

3.3.3 Ανάλυση των Κλάσεων που Χρησιμοποιήθηκαν

Η χρήση κλάσεων στην εφαρμογή μας αποτελεί ένα ακόμη βήμα προς την λογική του εύκολα συντηρήσιμου κώδικα. Χρησιμοποιώντας ξεχωριστά αντικείμενα για διαφορετικές λειτουργίες με απλές μεθόδους για την εκτέλεση συγκεκριμένης εργασίας έχουμε διαχωρίσει περίπλοκα προβλήματα σε ένα σύνολο απλών. Οι κλάσεις λοιπόν που αναπτύξαμε είναι

διαχωρισμένες με κριτήρια τις λειτουργίες που επιτελούν. Έχουμε επιλέξει την χρήση τεσσάρων κλάσεων, μία για την αλληλεπίδραση με την βάση δεδομένων, μία για την διατήρηση και εμφάνιση μηνυμάτων, μία για λειτουργίες λογαριασμού χρηστών και μία γενικού σκοπού.

Η πρώτη κλάση που εξετάζουμε ονομάζεται “Database” και βρίσκεται στο αρχείο “Database.php”. Η κλάση αυτή είναι υπεύθυνη για την δημιουργία και εκτέλεση ερωτημάτων στην βάση δεδομένων. Στην κλάση “Database” βρίσκουμε τα στοιχεία σύνδεσης στην βάση δεδομένων (Κώδικας 10, γραμμές 10-13). Έχοντας τα στοιχεία αυτά, κατά την μέθοδο δημιουργίας της κλάσης επιχειρούμε την σύνδεση με την βάση δεδομένων δημιουργώντας ένα αντικείμενο PDO που ονομάζουμε “$conn” (Κώδικας 10, γραμμές 15-19). Στην συνέχεια χρησιμοποιούμε το αντικείμενο αυτό για κάθε αλληλεπίδραση με την βάση.

```
1. <?php
2.
3. class Database {
4. // Database class variables
5. public $conn;
6.
7. // Class constractor function
8. //  initialise the PDO connection
9. function __construct(){
10.   $db_host="localhost"; // Host name
11.   $db_username="root"; // Mysql username
12.   $db_password=""; // Mysql password
13.   $db_name="vhdl_compiler"; // Database name
14.
15.   try{
16.     $this->conn = new PDO("mysql:host=$db_host;dbname=$db_name;char-
    set=utf8",$db_username,$db_password);
17.   }catch(PDOException $pe){
18.     die('Connection error:' . $pe->getmessage());
19.   }
20. }
```

Κώδικας 10: Database.php - Η μέθοδος κατασκευής της κλάσης Database

Αξίζει ενδεικτικά να εξετάσουμε μία από τις μεθόδους της κλάσης “Database” όπως την μέθοδο επιβεβαίωσης χρήστη. Η μέθοδος αυτή έχει ως είσοδο το όνομα και τον κωδικό που έδωσε ο χρήστης για την σύνδεσή του στην εφαρμογή. Ο κωδικός που δόθηκε μετασχηματίζεται από την συνάρτηση κατακερματισμού md5 καθώς συγκρατούμε με αυτή την μορφή τους κωδικούς χρηστών (Κώδικας 11, γραμμή 4). Στην συνέχεια δημιουργούμε το ερώτημα χρησιμοποιώντας όταν είναι δυνατό προετοιμασμένες δηλώσεις μεταβλητών καθώς προσθέτουν ένα επιπλέον επίπεδο ασφάλειας (Κώδικας 11, γραμμές 5-6). Έπειτα προχωράμε στην υποβολή του ερωτήματος και εάν γίνει επιτυχώς λαμβάνουμε την απόκριση και επιβεβαιώνουμε τον κωδικό (Κώδικας 11, γραμμές 8-10). Τέλος, εάν γίνει ταυτοποίηση του κωδικού με αυτόν που έδωσε ο χρήστης, επιστρέφεται το αναγνωριστικό “id” του χρήστη, αλλιώς επιστρέφεται 0 (Κώδικας 11, γραμμές 11 και 14).

```
1. // Confirm username and password
2. //  on success returns user ID, else 0
3. function confirm_user($username, $password) {
4.    $password = md5($password);
5.    $query = "SELECT * FROM users WHERE username = :username";
6.    $statement = $this->conn->prepare($query);
7.
8.    if( $statement->execute(array(':username'=>$username)) ){
9.      $result = $statement->fetch();
10.     if($result['password'] == $password){
11.       return $result['id'];
12.     }
13.   }
14.   return 0;
15. }
```

Κώδικας 11: Database.php - Η μέθοδος επιβεβαίωσης χρήστη

Βλέπουμε λοιπόν ότι οι μέθοδοι της κλάσης "Database" λειτουργούν, με συγκεκριμένες εισόδους, την εκτέλεση του αντίστοιχου ερωτήματος και συνήθως μία μικρή επεξεργασία των αποτελεσμάτων για την προσφορά συγκεκριμένων εξόδων.

```
1. class Messages {
2.    // Messages array
3.    //success, info, warning, danger
4.    $messages=array(
5.      "en" => array(
6.        "fail_login" => array("The Username or Password were
   wrong.","danger"),
7.        "success_login" => array("Logged in successfully.","success"),
8.        [...]
9.       ),
10.     "gr" => array(
11.       "fail_login" => array("Το όνομα ή ο κωδικός που δώσατε είναι λά-
   θος.","danger"),
12.       "success_login" => array("Επιτυχής είσοδος χρήστη.","success"),
13.       [...]
14.     )
15.   );
16.
17.   // Text array
18.   $text=array(
19.     "en" => array(
20.       "home_header" => "HDL Everywhere.",
21.        [...]
22.     ),
23.     "gr" => array(
24.       "home header" => "HDL Παντού.",
25.        [...]
26.     )
27.   );
```

Κώδικας 12:Messages.php - Οι πίνακες μηνυμάτων και κειμένου της κλάσης Messages

Η δεύτερη κλάση που θα δούμε είναι η "Messages" στο αρχείο "Messages.php". Η κλάση αυτή έχει ως βασικό στόχο την διατήρηση και κατηγοριοποίηση των μηνυμάτων και του κειμένου που βρίσκεται στις σελίδες της εφαρμογής μας. Αυτό επιτυγχάνεται χρησιμοποιώντας

δύο πίνακες, έναν για κάθε κατηγορία. Αρχικά, ο πίνακας που συγκρατεί τα μηνύματα ανάδρασης ονομάζεται "$messages" και διαχωρίζεται σε υποπίνακες με βάση την γλώσσα (Κώδικας 12,γραμμές 4-15). Ο κάθε υποπίνακας γλώσσας περιέχει όλα τα μηνύματα στην αντίστοιχη μετάφραση διευκολύνοντας την προσθήκη νέας γλώσσας. Τα μηνύματα αποτελούνται από ένα αναγνωριστικό κλειδί και έναν πίνακα όπου συγκρατούμε το μήνυμα και τον τύπο εμφάνισής του (Κώδικας 12, γραμμές 11-12). Οι τύποι εμφάνισης που υποστηρίζονται βασίζονται στις διαφορετικές κλάσεις "alert" που χρησιμοποιεί το Bootstrap και διαχωρίζονται στις κατηγορίες "success" με πράσινο χρώμα, "info" με μπλε χρώμα, "warning" με πορτοκαλί χρώμα και "danger" με κόκκινο χρώμα. Με παρόμοιο τρόπο διατηρούμε τον πίνακα κειμένων "$text" όπου παρατηρούμε ως μόνη διαφορά την έλλειψη τύπου εμφάνισης (Κώδικας 12, γραμμές 18-27). Ο τύπος εμφάνισης δεν είναι απαραίτητος στο κείμενο αφού η εμφάνισή του αντιμετωπίζεται ξεχωριστά σε κάθε περίπτωση από το θέμα εμφάνισης.

```
1. // Display messages
2. function display_msg($lang, $msg_codes_array) {
3.   foreach($msg_codes_array as $msg_id){
4.     echo '<div class="alert alert-'.$this->messages
   [$lang][$msg_id][1].'">'
5.          .'<a href="#" class="close" data-dismiss="alert" aria-la-
   bel="close">&times;</a>'
         .$this->messages[$lang][$msg_id][0]

6.          .'</div>';
7.   }
8.   $_SESSION['vhdl_msg']=array();
9. }
```

Κώδικας 13: Messages.php - Μέθοδος εμφάνισης μηνυμάτων

Συγκεκριμένα για την εμφάνιση μηνυμάτων, η κλάση "Messages" περιέχει την μέθοδο "display_smg". Η μέθοδος αυτή δέχεται ως εισόδους την επιλεγμένη γλώσσα και έναν πίνακα με τα κλειδιά των μηνυμάτων για εμφάνιση (Κώδικας 13, γραμμή 2). Κάθε μήνυμα εμφανίζεται, στην κατάλληλη γλώσσα, σε ένα στοιχείο "div" που χρησιμοποιεί την Bootstrap κλάση "alert" με τον κατάλληλο τύπο (Κώδικας 13, γραμμές 3-7). Έπειτα αδειάζουμε την συνεδρία "$_SESSION['vhdl_msg']" όπου εισάγονται τα μηνύματα προς εμφάνιση σε κάθε εκτέλεση σεναρίου (Κώδικας 13, γραμμή 8). Έτσι έχουμε την κλάση "Messages" που προσφέρει την διατήρηση και εμφάνιση μηνυμάτων ανάδρασης και κειμένου. Η κλάση αυτή μας επιτρέπει να επέμβουμε εύκολα και άμεσα στα διάφορα μηνύματα είτε πρόκειται για αλλαγή τύπου είτε για προσθήκη μηνύματος ή ακόμη και προσθήκη γλώσσας.

Η επόμενη κλάση που θα αναφέρουμε ονομάζεται "User" και βρίσκεται στο αρχείο "User.php". Η κλάση αυτή προσφέρει λειτουργίες που χρησιμοποιούν τα στοιχεία του χρήστη για την εξαγωγή αποτελεσμάτων.

```
1. <?php
2. class User {
3.   // User class variables
4.   $username,$id,$logged_in,$type;

5.   // Class constractor function
6.   //  initialise the user variables through the session
7.   function __construct($vhdl_user){
8.     $this->username = $vhdl_user['username'];
9.     $this->id = $vhdl_user['id'];
10.    $this->logged_in = $vhdl_user['loged_in'];
11.    $this->type = $vhdl_user['type'];
12.  }
```

Κώδικας 14: User.php - Ορισμός μεταβλητών και η μέθοδος κατασκευής της κλάσης User

Η κλάση "User" συγκρατεί πληροφορίες για τον χρήστη με παρόμοιο τρόπο που βρίσκονται στην συνεδρία "$_SESSION['vhdl_user']". Έτσι, κατά την δημιουργία ενός αντικειμένου δέχεται ως είσοδο τον πίνακα αυτής της συνεδρίας και μεταφέρει τα στοιχεία του σε αντίστοιχες μεταβλητές της κλάσης (Κώδικας 14). Χρησιμοποιούμε την μεταφορά αυτής της πληροφορίας καθώς είναι πιθανό μέσα σε ένα σενάριο εκτέλεσης να χρειαστεί η επέμβαση στις πληροφορίες ενός χρήστη, χωρίς όμως την ανάγκη για συγκράτηση των αλλαγών σε μόνιμη βάση. Εκτός από την συγκράτηση των στοιχείων κάθε χρήστη, η κλάση αυτή μας προσφέρει και κάποιες μεθόδους για την εξαγωγή πληροφορίας από αυτά τα στοιχεία.

```
1. // Return true if the the user is the owner of the project, else return
   false
2. function validate_ownership($editors){
3.   if($this->type=='1'){
4.     $valid = true;
5.   }else{
6.     $valid = false;
7.     foreach($editors as $editor){
8.       if( $editor['username'] == $this->username && $editor['user_type']
   == 1){
9.         $valid = true;
10.      }
11.    }
12.  }
13.  return $valid;
14. }
```

Κώδικας 15: User.php - Η μέθοδος επαλήθευσης ιδιοκτησίας ενός έργου από τον χρήστη

Η πρώτη από αυτές τις μέθοδος αποσκοπεί στην επαλήθευση ιδιοκτησίας ενός έργου από τον χρήστη. Η μέθοδος αυτή έχει ως είσοδο έναν πίνακα με ονόματα συγγραφέων του έργου και του τύπου δικαιωμάτων σε αυτό το έργο για τον κάθε συγγραφέα (Κώδικας 15, γραμμή 2). Εάν ο χρήστης είναι τύπου διαχειριστή, τότε θεωρείται ιδιοκτήτης κάθε έργου και επιστρέφουμε αληθές (Κώδικας 15, γραμμές 3-4). Σε διαφορετική περίπτωση, βρίσκουμε την καταχώριση στον πίνακα που δόθηκε για τον χρήστη που εξετάζουμε και επιβεβαιώνουμε εάν έχει δικαιώματα ιδιοκτήτη στο συγκεκριμένο έργο (Κώδικας 15, γραμμές 7-11).

```
1. // Return true if the the user is an editor to the project
2. function validate_edit_rights($editors){
3.   if($this->type=='1'){
4.     $valid = true;
5.   }else{
6.     $valid = false;
7.     foreach($editors as $editor){
8.       if( $editor['username'] == $this->username){
9.         $valid = true;
10.      }
11.    }
12.  }
13.  return $valid;
14. }
```

Κώδικας 16: User.php - Η μέθοδος επαλήθευσης δικαιωμάτων συγγραφέα

Η δεύτερη μέθοδος αποσκοπεί στην επαλήθευση δικαιωμάτων συγγραφέα του χρήστη σε ένα έργο. Έχει τον ίδιο πίνακα ως είσοδο και λειτουργεί με τον ίδιο τρόπο με την διαφορά ότι ελέγχουμε εάν ο χρήστης υπάρχει στον πίνακα χωρίς να εξετάζουμε τα συγκεκριμένα δικαιώματα (Κώδικας 16). Έτσι επιβεβαιώνουμε ότι ο χρήστης είναι συγγραφέας και επιστρέφουμε αληθές ή ψευδές αντίστοιχα. Όπως γίνεται φανερό η κλάση "User" μας προσφέρει κάποιες χρήσιμες λειτουργίες αλλά έχει σχεδιαστεί κυρίως για την υποστήριξη ενός πολυπλοκότερου συστήματος χρηστών. Στοχεύει στην διευκόλυνση πιθανής μελλοντικής επέκτασης στον τρόπο που αντιμετωπίζεται κάθε χρήστης και στην διεύρυνση των δικαιωμάτων πρόσβασης σε διάφορες λειτουργίες από διαφορετικά επίπεδα χρηστών.

Τέλος, βλέπουμε την κλάση "General" που βρίσκεται στο αρχείο "General.php". Η κλάση αυτή αποτελείται από ένα σύνολο λειτουργιών γενικής χρήσης που δεν κατηγοριοποιούνται αλλού. Ένα αντικείμενο αυτής της κλάσης αποτελεί ουσιαστικά μία βιβλιοθήκη επέκτασης προσαρμοσμένη για την εφαρμογή που αναπτύξαμε. Ενδεικτικά, κάποιες από τις μεθόδους που προσφέρει η κλάση "General" είναι η "extract_file" που εξάγει αρχεία από ένα συμπιεσμένο αρχείο και δημιουργεί τις κατάλληλες σειρές στην βάση δεδομένων (Κώδικας 17, γραμμές 1-23), η "send_email" που δέχεται το μήνυμα, τίτλο και διεύθυνση e-mail ως είσοδο αναλαμβάνει την αποστολή του αντίστοιχου e-mail (Κώδικας 17, γραμμές 26-29) και η "create_short_code" που μετατρέπει μία συμβολοσειρά σε έγκυρο όνομα αρχείου ή καταλόγου (Κώδικας 18).

```
1. function extract_file($file, $directory, $project_id){
2.   $zip = new ZipArchive;
3.   $db = new Database;
4.
5.   if ($zip->open($file) === true) {
6.     for($i = 0; $i < $zip->numFiles; $i++) {
7.       $filearray = $zip->getNameIndex($i);
8.       $fileinfo = pathinfo($filearray);
9.       $absolute_dir = $directory;
10.
11.      if( isset( $fileinfo['extension'] ) ){
12.        if( $db->add_file($fileinfo['basename'], $project_id) >0 ){
13.          copy("zip://".$file."#".$filearray,
    $absolute_dir."/".$fileinfo['basename']);
14.        }else{
15.          return false;
16.        }
17.      }
18.    }
19.    $zip->close();
20.    return true;
21.  }
22.  return false;
23. }
24.  [...]
25. // Send Email
26. function send_email($message, $subject, $mail){
27.  $headers = 'From: noreply@spam.vlsi.gr'."\r\n".'Reply-To: nore-
    ply@spam.vlsi.gr'."\r\n".'X-Mailer: PHP/'.phpversion();
28.  return mail($mail, $subject, $message, $headers);
29. }
```

Κώδικας 17: General.php - Οι μέθοδοι extract_file και send_email

```
1. // Return the shortcode of a string
2. function create_short_code($string){
3.   $replace_pairs = array(
4.     " " => "-",
5.     "α" => "a",
6.     "β" => "b",
7.     [...]
8.   );
9.   $string = trim($string," ");
10.  $string = strtr($string, $replace_pairs);
11.  $string = strtolower($string);
12.  $string = preg_replace("/[^a-z0-9\-\.]/","",$string);
13.  $string = preg_replace("/-+/","-",$string);
14.  return $string;
15. }
```

Κώδικας 18: General.php - Η μέθοδος create_short_code

Γίνεται λοιπόν φανερό ότι η κλάση “General” είναι ένα δυνατό εργαλείο της εφαρμογής μας που προσφέρει ένα σύνολο λειτουργιών πολλαπλών χρήσεων. Είναι φυσικά πιθανό κάποιες λειτουργίες που προσφέρονται από την κλάση “General” να δημιουργήσουν μέσα από μελλοντικές επεκτάσεις ξεχωριστές κατηγορίες και να αποτελέσουν διαφορετικές κλάσεις.

Είδαμε τις τέσσερις κλάσεις που χρησιμοποιεί η εφαρμογή μας αναλυτικά. Μέσα από την ανάλυσή τους έγινε ξεκάθαρη η λειτουργία κάθε κλάσης και την κατηγορία λειτουργιών που της αντιστοιχούν. Εξηγήσαμε τον τρόπο και την λογική σχεδιασμού αντιπροσωπευτικών μεθόδων και είναι πλέον ευδιάκριτος ο τρόπος που η χρήση αυτών των κλάσεων μας εξυπηρετεί τόσο στην παρούσα κατάσταση της εφαρμογής όσο και σε πιθανές επεκτάσεις. Στο επόμενο υποκεφάλαιο θα δούμε το περιβάλλον εμφάνισης και τον τρόπο που οι διεργασίες της εφαρμογής μας προσφέρονται στον χρήστη.

3.3.4 Υλοποίηση και Χρήση του Θέματος Εμφάνισης

```
1. <!DOCTYPE html>
2. <html>
3. <head>
4. <title><?php $messages->text[$_SESSION['vhdl_lang']]['title'] ?></title>
5.
6.   <meta charset="utf-8">
7.   <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
8.   <meta http-equiv="Content-Style-Type" content="text/css" />
9.   <meta http-equiv="Content-Script-Type" content="text/javascript" />
10.  <meta name="viewport" content="width=device-width, initial-scale=1">
11.
12.  <link rel="stylesheet" type="text/css" href="<?php echo $BASE_URL
   ?>/theme/bootstrap/bootstrap.min.css">
13.  <link rel="stylesheet" type="text/css" href="<?php echo $BASE_URL
   ?>/theme/bootstrap/bootstrap-theme.min.css">
14.  <link rel="stylesheet" type="text/css" href="<?php echo $BASE_URL
   ?>/theme/css/style.css" media="screen" />
15.
16.  <script type="text/javascript">window.base_url="<?php echo $BASE_URL;
   ?>";</script>
17.  <script type="text/javascript" src="https://code.jquery.com/jquery-
   2.2.0.min.js"></script>
18.
19. </head>
20.
21. <body>
22.  <?php require('nav-bar.php'); ?>
23.  <div id="general-popup" class="hidden"></div>
24.  <div class="container">
```

Κώδικας 19: header.php - Ο κώδικας του αρχείου

Το θέμα εμφάνισης αποτελείται από ένα σύνολο αρχείων που περιγράφουν τον τρόπο που οι σελίδες της εφαρμογής παρουσιάζονται στον χρήστη. Χρησιμοποιούμε τόσο HTML και CSS για την περιγραφή και μορφοποίηση των στοιχείων όσο και την γλώσσα JavaScript για την προσθήκη δυναμικών λειτουργιών πάνω σε αυτά τα στοιχεία. Το θέμα εμφάνισης δημιουργεί

ένα πλέγμα γύρω από τις εξόδους λειτουργιών του εξυπηρετητή και προσαρμόζει τα στοιχεία τους ώστε να εμφανίζονται με συγκεκριμένο τρόπο. Η ανάπτυξη του θέματος εμφάνισης βασίζεται στο πλαίσιο Bootstrap όπως παρουσιάστηκε στο θεωρητικό υπόβαθρο. Από το πλαίσιο Bootstrap λαμβάνουμε χρήσιμες κλάσεις CSS και συναρτήσεις JavaScript που προσαρμόσαμε για τις ανάγκες του δικού μας θέματος. Οι βάσεις του θέματος εμφάνισης βρίσκονται σε τρία αρχεία και περιβάλλουν κάθε σελίδα που επισκέπτεται ο χρήστης.

Το πρώτο αρχείο ονομάζεται “header.php” και περιλαμβάνει τα πρώτα HTML στοιχεία που χρησιμοποιεί η κάθε σελίδα. Τα στοιχεία αυτά βρίσκονται κυρίως μεταξύ των ετικετών τίτλου και περιλαμβάνουν τον τίτλο της σελίδας, την κωδικοποίηση μετα-στοιχεία και την εισαγωγή των αρχείων CSS και απαραίτητων αρχείων JavaScript (Κώδικας 19, γραμμές 4-19). Επίσης στο αρχείο “header.php” ξεκινάμε την ετικέτα σώματος και εισάγουμε το επόμενο βασικό αρχείο θέματος, το “nav-bar.php” πριν την δημιουργία του στοιχείου εμφάνισης μηνυμάτων ανάδρασης και την αρχή του στοιχείου που θα περιέχει την έξοδο των λειτουργιών (Κώδικας 19, γραμμές 22-24).

```
1. <nav class="navbar navbar-default navbar-top">
2.   <div class="container-fluid">
3.     <div class="collapse navbar-collapse pull-left" id="navbar-header">
4.       <ul class="nav navbar-nav">
5.         <li class="dropdown">
6.           <a id="search-nav-dropdown" href="javascript:void(0)"
   class="dropdown-toggle" data-toggle="dropdown">Projects<span
   class="caret"></span></a>
7.           <ul id="search-nav-dropdown-ul" class="dropdown-menu">
8.             [...]
9.           </ul>
10.        </li>
11.        <li class="navbar-form">
12.          <input type="hidden" id="nav-search-type" value="Projects">
13.          <input type="text" class="form-control nav-search-typeahead"
   autocomplete="off" placeholder="Search" id="nav-search">
14.        </li>
15.      </ul>
16.    </div>
17.    <div class="navbar-user pull-right">
18.      [...]
19.    </div>
20.  </div>
21. </nav>
```

Κώδικας 20: nav-bar.php - Η γενική δομή της μπάρας εργαλείων με τις κλάσεις που προσφέρει το πλαίσιο Bootstrap

Στο αρχείο “nav-bar.php” γίνεται χρήση των διαθέσιμων εργαλείων του Bootstrap για την δημιουργία μίας μπάρας εργαλείων στην κορυφή της σελίδας. Ακολουθούμε την δομή που συνιστάται για την δημιουργία μίας τέτοιας μπάρας που περικλείει, με την κλάση “container-fluid”, τα στοιχεία για την πλήρη οριζόντια κάλυψη της σελίδας. Έπειτα διαχωρίζονται τα στοιχεία ταξινομημένα στην αριστερή και στην δεξιά πλευρά της μπάρας με τις κλάσεις “pull-left” και “pull-right” αντίστοιχα (Κώδικας 20). Χρησιμοποιούμε την αριστερή πλευρά της μπάρας εργαλείων για την διατήρηση μίας εσωτερικής μηχανής αναζήτησης έργων, χρηστών

και εξαρτημάτων. Η δεξιά πλευρά της μπάρας εργαλείων περιέχει διαφορετικές λειτουργίες ανάλογα με τον τύπο του επισκέπτη. Εάν ο επισκέπτης δεν είναι συνδεμένος σε έναν λογαριασμό χρήστη, η λειτουργία που παρουσιάζεται είναι ένας σύνδεσμος για την αρχική σελίδα. Εάν υπάρχει συνδεμένος χρήστης, προσφέρονται σύνδεσμοι για την αρχική σελίδα του χρήστη, την επεξεργασία του λογαριασμού του καθώς και η επιλογή για αποσύνδεση. Εάν ο λογαριασμός είναι τύπου διαχειριστή, προσφέρονται οι λειτουργίες χρήστη και επιπλέον ένας σύνδεσμος για την σελίδα διαχείρισης. Έτσι προσφέρουμε κάποιες βασικές λειτουργίες σε κάθε επισκέπτη στην αρχή κάθε σελίδας όπου βρίσκονται σε ένα εύκολα προσβάσιμο και αντιληπτό σημείο χωρίς την παρεμπόδιση και εμπλοκή τους με την υπόλοιπη σελίδα.

```
1.        <br /><br /><br />
2.        <div class="footer">
3.          <div id="copyright">
4.            <center>
5.              Copyright 2014-2016<br/>
6.              Minas Dasygenis, <a href="http://arch.icte.uowm.gr/mdasy-
   genis">http://arch.icte.uowm.gr/mdasygenis</a>.<br />
7.              Lymperidis Efstathios, <a
   href="https://gr.linkedin.com/in/efstathios-lymperidis-
   98b5747b">https://gr.linkedin.com/in/efstathios-lymperidis-
   98b5747b</a>.<br />
8. Licensed under the Apache License, Version 2.0 (the "License").<br />
9. http://www.apache.org/licenses/LICENSE-2.0<br />
10.            </center>
11.          </div>
12.          <br />
13.          <div id="download">
14.            <center>The source code is available [<a
   href="https://github.com/Stathislyb/hdl-compiler">Here</a>].</center>
15.          </div>
16.        </div>
17.      </div>
18.
19.    </div><!--end of container-->
20.    <script type="text/javascript" src="<?php echo $BASE_URL
   ?>/theme/js/functions.js"></script>
21.    <script type="text/javascript" src="<?php echo $BASE_URL
   ?>/theme/bootstrap/bootstrap.min.js"></script>
22.    <script type="text/javascript" src="<?php echo $BASE_URL
   ?>/theme/js/bootstrap3-typeahead.min.js"></script>
23.    <script type="text/javascript" src="<?php echo $BASE_URL
   ?>/theme/js/validator.js"></script>
24.  </body>
25. </html>
```

Κώδικας 21: footer.php - Ο κώδικας του αρχείου

Το τελευταίο αρχείο που αποτελεί τις βάσεις του θέματος εμφάνισης είναι το “footer.php”. Το αρχείο αυτό εισάγεται στο τέλος της σελίδας και αποτελείται από δυο μέρη. Το μέρος που περιέχει τις πληροφορίες ιδιοκτησίας (Κώδικας 21, γραμμές 2-17) και το μέρος που κλείνουμε τις ετικέτες που περιλαμβάνουν την σελίδα. Στο τελευταίο εισάγουμε τα αρχεία JavaScript που δεν είναι απαραίτητο να μεταφορτωθούν στο “header.php” (Κώδικας 21, γραμμές 19-25). Τα

αρχεία JavaScript που εισάγουμε στο “header.php” είναι απαραίτητα για την σωστή διαμόρφωση στοιχείων μέσα στην σελίδα. Αντίθετα, αυτά που εισάγουμε στο “footer.php” έχουν λειτουργίες που χρειάζονται προ απαιτούμενα στοιχεία. Επίσης αυτή τη τεχνική επιτρέπει σε μερικούς περιηγητές να εμφανίσουν την σελίδα πριν ακόμη ολοκληρώσουν την μεταφόρτωση των επιπλέον αρχείων JavaScript.

```
1.  //Add Event listeners when the document is fully loaded
2.  $( document ).ready(function() {
3.
4.    // Handle Navbar Search type change
5.    $("#search-nav-dropdown-ul li a").click(function(){
6.      [...]
7.    });
8.
9.    $('#select_all').click(function(event) {
10.      [...]
11.   });
12.
13.   [...]
14.
15. });
16.
17. function navbar_search_project(e) {
18.   [...]
19. };
20.
21. function navbar_search_user(e) {
22.   [...]
23. };
24.
25. [...]
```

Κώδικας 22: functions.js - Η γενική δομή του αρχείου

Εκτός των τριών βασικών αρχείων του θέματος εμφάνισης, έχουμε και ένα αρχείο που αναλαμβάνει την εισαγωγή λειτουργιών στα στοιχεία που εμφανίζονται. Το αρχείο αυτό ονομάζεται “function.js” και αποτελείται από δύο μέρη. Το πρώτο μέρος περιλαμβάνεται σε μία συνάρτηση που προσφέρει η βιβλιοθήκη jQuery και επικυρώνει ότι ο κώδικας θα εκτελεστεί αφού έχει ολοκληρωθεί η φόρτωση της σελίδας (Κώδικας 22, γραμμή 2). Εσωτερικά της συνάρτησης αυτής έχουμε τις δηλώσεις αντιμετώπισης γεγονότων (Κώδικας 22, γραμμές 4-13). Είναι απαραίτητο να επιβεβαιώσουμε την ύπαρξη των στοιχείων HTML γιατί σε διαφορετική περίπτωση η κάθε δήλωση αντιμετώπισης γεγονότων θα οδηγούσε σε αποτυχία αφού δεν υπάρχει το στοιχείο που θέλουμε να συνδέσουμε με την αντίστοιχη αντιμετώπιση. Το δεύτερο μέρος του αρχείου “functions.js” αποτελείται από ανεξάρτητες συναρτήσεις. Οι συναρτήσεις αυτές χρησιμοποιούνται συχνά στην αντιμετώπιση γεγονότων και αποτελούν χρήσιμα πακέτα λειτουργιών που πολλές φορές βρίσκουν εφαρμογή σε διαφορετικά τμήματα κώδικα.

Παραπάνω παρουσιάσαμε τα βασικά στοιχεία που συντελούν το θέμα εμφάνισης της εφαρμογής μας. Τα στοιχεία αυτά χρησιμοποιούν εργαλεία όπως το πλαίσιο Bootstrap και τον κειμενογράφο Ace προσαρμόζοντάς τα στο δικό μας περιβάλλον αλληλεπίδρασης. Ταυτόχρονα όμως εισάγουν τις δικές τους λειτουργίες και αποτελούν ένα σύστημα που περιβάλλει τα αποτελέσματα διαδικασιών του εξυπηρετητή.

3.3.5 Τεχνικές Υλοποίησης των Θεμάτων Ασφάλειας

Στα προηγούμενα υποκεφάλαια αναλύσαμε τους τρόπους που επιτυγχάνουμε τις λειτουργίες που συντελούν την εφαρμογή μας τόσο στον εξυπηρετητή όσο και στον χρήστη. Στην συνέχεια θα εξηγήσουμε τον τρόπο που αναπτύξαμε ένα σύνολο διαδικασιών και τεχνικών, συνυφασμένα στις παραπάνω λειτουργίες με σκοπό την εισαγωγή διαφόρων επιπέδων ασφάλειας.

Η πρώτη επαφή με την ασφάλεια της σελίδας γίνεται κατά την φόρτωση. Όπως είδαμε στο αρχείο "loader.php" καλείται το αρχείο "security.php" πριν τους διαχειριστές αλληλεπίδρασης. Το "security.php" λειτουργεί ως φίλτρο επιβεβαιώνοντας ότι τα δεδομένα που αποστέλνονται στο εξυπηρετητή με την μέθοδο POST δεν περιέχουν απειλές προς την ασφάλεια της εφαρμογής.

```
1. <?php
2.
3. $security_breach = false;
4.
5. // Sanitize string POST data
6. $string_array = array('ajax_action','username', [...]);
7. foreach($string_array as $string_element){
8.    if( isset($_POST[$string_element]) ){
9.      $security_temp = filter_var($_POST[$string_element], FILTER_SANI-
   TIZE_STRING);
10.     if( $_POST[$string_element] != $security_temp ){
11.       $security_breach = true;
12.     }
13.   }
14. }
```

Κώδικας 23: security.php - Αρχικοποίηση και έλεγχος γενικών δεδομένων συμβολοσειρών

Το αρχείο "security.php" αρχικοποιείται με την υπόθεση ότι δεν υπάρχει παραβίαση ασφάλειας (Κώδικας 23, γραμμή 3). Έπειτα προχωράμε στην έρευνα αυτής της υπόθεσης ελέγχοντας τα πιθανά δεδομένα POST που χρησιμοποιούν οι λειτουργίες μας. Αρχικά ελέγχουμε τα δεδομένα γενικής μορφής συμβολοσειρών συγκρατώντας τις πιθανές ονομασίες αυτών σε έναν πίνακα και ελέγχοντας την ύπαρξή τους στον πίνακα "$_POST[]" (Κώδικας 23, γραμμές 6-8). Εάν εντοπίσουμε τη ύπαρξη τιμής σε μία από αυτές τις θέσεις του πίνακα, επιχειρούμε την απόσταξή της από ένα φίλτρο συγκεκριμένα για τιμές συμβολοσειρών (Κώδικας 23, γραμμή 9). Χρησιμοποιούμε μία προσωρινή τιμή για το αποτέλεσμα του φίλτρου και ελέγχουμε εάν υπάρχουν αλλαγές όπου θα σημαίνουν την αποστολή μη έγκυρων δεδομένων και πιθανή παραβίαση ασφάλειας (Κώδικας 23, γραμμές 10-11). Με παρόμοιο τρόπο ελέγχουμε στην συνέχεια τα δεδομένα γενικού τύπου ακέραιων αριθμών όπου χρησιμοποιούμε το αντίστοιχο φίλτρο.

```
1. if( isset($_POST['selected_ids']) ){
2.    $selected = explode('-',$_POST['selected_ids']);
3.    foreach ($selected as $file_id) {
4.      $file_id_sanitized = filter_var($file_id, FILTER_SANITIZE_NUM-
   BER_INT);
5.      if($file_id_sanitized != $file_id){
6.        $security_breach = true;
7.      }
8.    }
9. }
10. if( isset($_POST['email_edit'])) {
11.   $security_temp =  filter_var($_POST["email_edit"], FILTER_VALI-
   DATE_EMAIL);
12.   if( $_POST['email_edit'] != $security_temp || strlen(
   $_POST["email_edit"]) >50 ){
13.     $security_breach = true;
14.     array_push($_SESSION['vhdl_msg'], 'invalid_mail');
15.   }
16. }
```

Κώδικας 24: security.php - Έλεγχος ασφάλειας για τα δεδομένα selected_ids και email_edit

Υπάρχουν όμως περιπτώσεις όπου οι τιμές των δεδομένων δεν ακολουθούν γενικές μορφές αλλά έχουν συγκεκριμένη μορφοποίηση. Τέτοιες περιπτώσεις είναι για παράδειγμα τα δεδομένα “selected_ids” και “email_edit”. Στην πρώτη περίπτωση τα δεδομένα αποτελούνται από ακέραιους αριθμούς χωρισμένα με παύλες. Έτσι επιχειρούμε πρώτα την διάσπαση των αριθμών και προχωρούμε μετά σε ξεχωριστό έλεγχο για τον καθένα με τον τρόπο που ελέγχουμε την γενική μορφή ακέραιων αριθμών (Κώδικας 24, γραμμές 1-9). Στην δεύτερη περίπτωση έχουμε τον επιπλέον περιορισμό των πενήντα χαρακτήρων. Η εξέταση των δεδομένων γίνεται όπως τις γενικής μορφής συμβολοσειρών με την προσθήκη ελέγχου του αριθμού χαρακτήρων όπου εάν το επιτρεπόμενο μέγεθος ξεπεραστεί, θεωρείτε παραβίαση ασφάλειας (Κώδικας 24, γραμμές 10-16).

```
1. if( $security_breach ){
2.   array_push($_SESSION['vhdl_msg'], 'invalid_POST');
3.   header("Refresh:0");
4.   exit();
5. }
```

Κώδικας 25: security.php - Αντιμετώπιση πιθανής παραβίασης ασφάλειας

Αφού ολοκληρώσουμε όλους του ελέγχους, προχωρούμε στην εξέταση της αρχικής υπόθεσης. Σε περίπτωση που δεν έχει βρεθεί κάποια πιθανή παραβίαση ασφάλειας, επιτρέπουμε την συνέχεια της ροής εκτέλεσης κανονικά. Αλλιώς δημιουργούμε μήνυμα ανάδρασης ώστε να ενημερώσουμε τον χρήστη, προχωρούμε στην επαναφόρτωση της σελίδας και σταματάμε την εκτέλεση σεναρίων PHP (Κώδικας 25, γραμμές 1-5). Έτσι ολοκληρώνεται το πρώτο επίπεδο ασφάλειας που στοχεύει στην αντιμετώπιση αποστολής μη επιτρεπόμενων τύπων δεδομένων στον εξυπηρετητή.

Το δεύτερο επίπεδο ασφάλεια βρίσκεται κατά τις αλληλεπιδράσεις της εφαρμογής με την βάση δεδομένων. Έχουμε είναι αναφερθεί επιφανειακά στον τρόπο υλοποίησης αυτού του επιπέδου αλλά στην συνέχεια θα το ερευνήσουμε ποιο διεξοδικά. Το επίπεδο αυτό αποτελείται

κατά κόρον από την χρήση του αντικειμένου PDO για την αποστολή ερωτημάτων στην βάση δεδομένων.

```
1. // Select the user's information by search parameter
2. public function get_user_information($user, $search_type) {
3.   $query = "SELECT * FROM users WHERE  ".$search_type." = :user ";
4.   $statement = $this->conn->prepare($query);
5.   $statement->execute(array('user' => $user));
6.   return $statement->fetch();
7. }
```

Κώδικας 26: Database.php - Η μέθοδος γενικής αναζήτησης χρήστη από την βάση δεδομένων

Για την διευκόλυνση της εξέτασης του επιπέδου αυτού θα χρησιμοποιήσουμε ως παράδειγμα την μέθοδο "get_user_information" που δέχεται ως εισόδους ένα στοιχείο του χρήστη και το είδος αυτού του στοιχείου ώστε να επιστρέψει όλα τα στοιχεία αυτού του χρήστη. Παρέχει ουσιαστικά μία γενική αναζήτηση χρήστη στην βάση δεδομένων. Λόγο όμως της γενικότητας, ο τύπος του στοιχείου αναζήτησης δεν είναι σταθερός κι έτσι δεν μπορεί να περάσει από σταθερό φίλτρο στο πρώτο επίπεδο ασφάλειας. Έτσι ο μόνος τρόπος για την διασφάλιση της ασφάλειας των δεδομένων αυτού του στοιχείου είναι η εισαγωγή τους στο ερώτημα SQL μέσω της προετοιμασίας από το αντικείμενο PDO. Η προετοιμασία γίνεται εισάγοντας ένα δείκτη μέσα στο ερώτημα όπως το ":user" (Κώδικας 26, γραμμή 3). Στην συνέχεια κάνουμε την προετοιμασία του ερωτήματος και το εκτελούμε αντιστοιχίζοντας τον δείκτη με την μεταβλητή (Κώδικας 26, γραμμές 4-5). Με αυτόν τον τρόπο διασφαλίζουμε ότι τα στοιχεία που θα αντικαταστήσουν τον δείκτη θα αντιμετωπιστούν από την SQL ως μία τιμή. Έτσι εάν γίνει για παράδειγμα η εισαγωγή στα στοιχεία του χρήστη μίας τιμής όπως "OR IF 1=1 DELETE FROM users", σε μία απλή εισαγωγή του ερωτήματος στην βάση δεδομένων, θα μπορούσε να έχει ως αποτέλεσμα την διαγραφή όλων των χρηστών. Με την τεχνική όμως που χρησιμοποιούμε εδώ, η τιμή αυτή θα περιοριστεί στην αντιμετώπισή της, αποκλειστικά ως τιμή του στοιχείου που δίνεται από το "$search_type" και όχι ως ένα συνεχές ερώτημα. Επιστρέφοντας στο αρχικό παράδειγμα της μεθόδου "get_user_information", μπορούμε πλέον να εξηγήσουμε τον λόγο που κάποιες μεταβλητές δεν μπορούν να προετοιμαστούν. Τέτοιες μεταβλητές, όπως η "$search_type", πρέπει να βρίσκονται στο ερώτημα πριν την προετοιμασία του ώστε να είναι δυνατός ο ορισμός του τύπου δεδομένων του στοιχείου που αντιπροσωπεύουν κατά την προετοιμασία. Αυτές οι μεταβλητές όμως περιορίζονται σε συγκεκριμένες τιμές που καθορίζονται από τα γνωρίσματα του πίνακα κι έτσι είναι δυνατός ένας αρκετά πιο συγκεκριμένος έλεγχος των τιμών τους. Γίνεται λοιπόν φανερό πως τα πρώτα δύο επίπεδα περιστρέφονται γύρω από την δημιουργία ενός τοίχου προστασίας τις βάσεως δεδομένων.

Το τρίτο επίπεδο που χρησιμοποιούμε στοχεύει αντίθετα στην εισαγωγή ασφάλειας στους λογαριασμούς χρηστών. Το επίπεδο αυτό βρίσκεται μέσα στις λειτουργίες χρηστών και επιβεβαιώνει ότι η κάθε λειτουργία είναι διαθέσιμη εφόσον ο επισκέπτης έχει τα κατάλληλα δικαιώματα. Στο αρχείο "index.php" είδαμε μία εφαρμογή αυτού του επιπέδου όπου γίνεται έλεγχος για συνδεμένο λογαριασμό χρήστη πριν την διάθεση συγκεκριμένων σελίδων. Αυτή είναι μία συχνή εφαρμογή του επιπέδου σε πολλές λειτουργίες. Το επίπεδο αυτό γενικά κάνει χρήση των μεταβλητών που βρίσκονται στην συνεδρία "$_SESSION['vhdl_user']" κατά την είσοδο ενός χρήστη. Αρχικά έχουμε την μεταβλητή "$_SESSION['vhdl_user']['loged_in']" που επιβεβαιώνει έχοντας τιμή, και συγκεκριμένα την τιμή "1", ότι ο χρήστης είναι συνδεμένος σε έγκυρο λογαριασμό. Ο έλεγχος της μεταβλητής αυτής χρησιμοποιείται για τον περιορισμό

λειτουργιών που δεν είναι διαθέσιμες σε μη συνδεμένους επισκέπτες. Η δεύτερη μεταβλητή είναι η "$_SESSION['vhdl_user']['activated']" που επισημαίνει την ενεργοποίηση του λογαριασμού μέσω του e-mail επιβεβαίωσης. Η μεταβλητή αυτή έχει την τιμή 1 εάν ο χρήστης έχει επιβεβαιώσει τον λογαριασμό του ή έχει συνδεθεί με "SID" λογαριασμό επισκέπτη και 0 εάν ο χρήστης δεν έχει επιβεβαιώσει τον λογαριασμό του. Έπειτα έχουμε την μεταβλητή "$_SESSION['vhdl_user']['type']" που παίρνει την τιμή 1 για λογαριασμούς τύπου διαχειριστή και 0 στους υπόλοιπους. Ο έλεγχος αυτής τις τιμής γίνεται κατά κόρον για την βεβαίωση ότι οι λειτουργίες διαχειριστών εκτελούνται από λογαριασμούς χρηστών με τα αντίστοιχα δικαιώματα. Τέλος, έχουμε την χρήση "$_SESSION['vhdl_user']['username']" και "$_SESSION['vhdl_user']['id']" που παίρνουν αντίστοιχα τις τιμές του ονόματος και του αναγνωριστικού "id" του χρήστη ενώ στην περίπτωση συνδεμένου επισκέπτη "SID" περιέχουν τις τιμές "Guest" και "0". Αυτό μας επιτρέπει συχνά να απομονώσουμε λειτουργίες για τους συνδεμένους χρήστες και για τους συνδεμένους επισκέπτες κάνοντας τον έλεγχο για τις συγκεκριμένες τιμές που δίνουμε στην σύνδεση επισκεπτών. Το επίπεδο αυτό είναι ενσωματωμένο στις περισσότερες λειτουργίες και συχνά χρησιμοποιούμε διαδοχικούς ελέγχους απομονώνοντας σταδιακά τις ποιο λεπτές λειτουργίες όπως αυτές που προσφέρονται σε διαχειριστές.

Το τελευταίο επίπεδο ασφάλειας αφορά τα αρχεία της εφαρμογής. Καθώς χρησιμοποιούμε συγκεκριμένες ροές κίνησης θέλουμε να περιορίσουμε την δυνατότητα επίσκεψης του χρήστη σε καταλόγους και αρχεία εκτός της ροής που επιτρέπουμε.

```
1. <?php header("location: //".$_SERVER["SERVER_NAME"]); ?>
```

Κώδικας 27: index.php - Κώδικας του αρχείο ανακατεύθυνσης

Έτσι χρησιμοποιούμε σε κάθε κατάλογο εκτός του αρχικού ένα αρχείο "index.php" με συγκεκριμένη δομή ώστε να ανακατευθύνει τον χρήστη στην αρχική διεύθυνση του εξυπηρετητή (Κώδικας 27). Έτσι αρχικά επιβεβαιώνουμε ότι ο επισκέπτης δεν έχει πρόσβαση απευθείας στους καταλόγους τις εφαρμογής.

```
1. <?php
2. if( !isset($db) ){
3.   header("location: //".$_SERVER["SERVER_NAME"]);
4.   exit();
5. }
6. ?>
```

Κώδικας 28: Κώδικας ανακατεύθυνση που βρίσκεται σε κάθε αρχείο σελίδας

Όσο αφορά τα αρχεία στον κατάλογο "/pages/" που αποτελούν τον κορμό εμφάνισης κάθε σελίδας, περιέχουμε έναν έλεγχο για την επιβεβαίωση της σωστής ροής κίνησης. Ο έλεγχος αυτός στοχεύει στην εύρεση του αντικειμένου που δημιουργούμε στο αρχείο "loader.php" από την κλάση "Database" και προκαλεί ανακατεύθυνση σε περίπτωση που το αντικείμενο αυτό δεν υπάρχει (Κώδικας 28). Έτσι αρχικά επιβεβαιώνουμε ότι ο χρήστης επισκέπτεται την σελίδα από την σωστή ροή κίνησης ενώ έχουμε το δευτερεύον πλεονέκτημα ανίχνευσης προβλημάτων στην σύνδεση με την βάση δεδομένων. Με το επίπεδο αυτό ασφαλίζουμε την σωστή ροή κίνησης.

Με τα παραπάνω επίπεδα ασφάλειας καλύπτουμε τις ανάγκες της εφαρμογής μας σε αυτόν τον τομέα. Προβλέπουμε τόσο κακόβουλες επιθέσεις όσο και λάθη χρήσης που θα οδηγούσαν σε προβληματική λειτουργία. Έτσι ολοκληρώνουμε την ανάλυση του κορμού της εφαρμογής μας, ενώ στο επόμενο υποκεφάλαιο θα εξηγήσουμε την υλοποίηση ενός εργαλείου που αναπτύξαμε και προσαρμόσαμε στις υπόλοιπες λειτουργίες μας για την σχεδίαση κυματομορφών.

3.4 Υλοποίηση του Εργαλείου Σχεδίασης Κυματομορφών

Ένα πολύ σημαντικό στοιχείο της εφαρμογής που αναπτύξαμε είναι η προσομοίωση των έργων. Για να έχει όμως νόημα η προσομοίωση χρειαζόμαστε κάποιον τρόπο προβολής των αποτελεσμάτων. Τα αποτελέσματα της προσομοίωσης από το GHDL αποθηκεύονται σε ένα αρχείο VCD (Value Change Dump, Ένδειξη Αλλαγής Αξίας) [27]. Το αρχείο αυτό αρχικά αναφέρει κάθε σήμα προσδιορίζοντας κάποιες σχετικές λεπτομέρειες και αναθέτει στο καθένα ένα αναγνωριστικό. Στο κυρίως σώμα του αρχείου αναφέρονται οι χρονικές τιμές στις οποίες υπάρχουν αλλαγές στην τιμή τουλάχιστον ενός σήματος και κάτω από τους χρόνους αυτούς δίνεται ανά σειρά το αναγνωριστικό του σήματος και η καινούρια τιμή του. Γίνεται λοιπόν φανερό ότι η προβολή αυτού του αρχείου ως αποτέλεσμα προσομοίωσης θα παρείχε τις πληροφορίες που χρειάζεται ο χρήστης με έναν εξαιρετικά δυσανάγνωστο τρόπο. Έτσι αναπτύξαμε το εργαλείο σχεδίασης κυματομορφών που επιτρέπει την προβολή των σημάτων και των αλλαγών τους στον χρόνο προσομοίωσης ως κυματομορφές. Παρακάτω θα αναλύσουμε τον τρόπο που αναπτύξαμε αυτό το εργαλείο ξεκινώντας από τις τεχνικές εξαγωγής και μεταφοράς των δεδομένων συνεχίζοντας στην υλοποίηση της βασικής λειτουργίας σχεδίασης των κυματομορφών και κλείνοντας με την ανάπτυξη των υπόλοιπων λειτουργιών που συνθέτουν το εργαλείο.

3.4.1 Εξαγωγή και Μεταφορά Δεδομένων

Το πρώτο βήμα που πρέπει να κάνουμε για την προβολή των αποτελεσμάτων προσομοίωσης είναι η εξαγωγή τους από το αρχείο VCD και η μετατροπή τους σε εύκολα επεξεργάσιμη μορφή τόσο από ένα σενάριο PHP όσο και από το πρόγραμμα JavaScript που τρέχει στον περιηγητή του χρήστη και υλοποιεί τις λειτουργίες του εργαλείου. Το βήμα αυτό γίνεται όταν ο χρήστης ζητήσει την προβολή των αποτελεσμάτων προσομοίωσης. Μέσω AJAX αποστέλλουμε στο αρχείο ajax_handler.php την αίτηση για ανάγνωση και μετάφραση του αντίστοιχου VCD αρχείου.

```
1.  $handle = fopen($vcd_file, "r");
2.  if($handle){
3.    while (($line = fgets($handle)) !== false) {
4.     // if the line is timescale definition, keep it to the time infor-
    mation
5.     if(  preg_match('/^  1 .*/',$line) === 1){
6.       // trim unnecessary parts and keep the timescale units
7.       $timescale = trim(str_replace('1',"",$line));
8.     }
9.
10.    // if the line is module definition, add that module to the path
11.    if(  preg_match('/^\$scope module .*/',$line) === 1){
12.      // trim unnecessary parts and keep the module's name
13.      $line = str_replace('$scope module ',"",$line);
14.      $module .= "_-_".trim(str_replace(' $end',"",$line));
15.    }
```

Κώδικας 29: ajax_handler.php - Κομμάτι κώδικα εξαγωγής δεδομένων από το VCD αρχείο

Η μετάφραση των αποτελεσμάτων παίρνει αρχικά την μορφή ενός πίνακα όπου συγκρατούμε για το κάθε σήμα την ονομασία του, το μήκος του, την οντότητα στην οποία ανήκει και τους χρόνους στους οποίους αλλάζει τιμή καθώς και την τιμή που παίρνει. Εκτός αυτών, συγκρατούμε και κάποια στοιχεία της ίδιας της προσομοίωσης όπως τον συνολικό χρόνο προσομοίωσης, τον αριθμό χρόνων όπου υπάρχει τουλάχιστον μία αλλαγή τιμής καθώς και την χρονική υποδιαίρεση που χρησιμοποιείται στους χρόνους που μας δίνονται. Η διαδικασία αυτή γίνεται εξετάζοντας κάθε γραμμή του αρχείου VCD και συγκρίνοντάς την με συγκεκριμένες αναμενόμενες μορφές που αντιστοιχούν σε δηλώσεις δεδομένων που μας ενδιαφέρουν (Κώδικας 29). Καθώς η σύνθεση του πίνακα αυτού μπορεί να είναι σχετικά χρονοβόρα φτάνοντας στα μερικά δευτερόλεπτα για πολύ μεγάλα αρχεία VCD, χρησιμοποιούμε ένα ενδιάμεσο αρχείο τύπου JSON (JavaScript Object Notation, Συμβολισμός Αντικειμένου JavaScript) [28]. Στο αρχείο αυτό συγκρατούμε τον πίνακα με τα αποτελέσματα προσομοίωσης σε μορφή JSON. Έτσι μπορούμε αρχικά να ελέγξουμε εάν υπάρχει αλλαγή στο αρχείο VCD από την τελευταία του μετάφραση και εάν δεν υπάρχει κάποια αλλαγή να πάρουμε απευθείας τα δεδομένα που χρειαζόμαστε από το ενδιάμεσο αρχείο JSON.

```
1. $vcd_timestamp = $path.filemtime($vcd_file);
2. [...]
3. if(file_exists($vcd_timestamp.'.json') ){
4.   [...]
5. }else{
6.   [...]
7.   $fp = fopen($vcd_timestamp.'.json', 'w') or die("Unable to open
   file!");
8.   fwrite($fp, json_encode($data));
9.   fclose($fp);
```

Κώδικας 30: ajax_handler.php - Ο έλεγχος και δημιουργία του ενδιάμεσου αρχείου JSON

Ο έλεγχος αυτός επιτυγχάνεται με την χρήση της PHP συνάρτησης filemtime() η οποία επιστρέφει τον χρόνο στον οποίο έγινε η τελευταία αλλαγή σε ένα αρχείο. Κατά την δημιουργία του ενδιάμεσου αρχείου JSON παίρνουμε τον χρόνο αυτό για το αντίστοιχο VCD και τον χρησιμοποιήσουμε ως ονομασία του JSON αρχείου. Έτσι, στην επόμενη εκτέλεση θα ξαναπάρουμε τον χρόνο τελευταίας αλλαγής του VCD αρχείου και θα ελέγξουμε αρχικά εάν υπάρχει διαθέσιμο JSON αρχείο με ονομασία τον χρόνο που ζητάμε οπότε προχωράμε

ανάλογα του αποτελέσματος (Κώδικας 30). Τέλος ο πίνακας μεταφράζεται σε μορφή JSON εάν δεν έχουμε πάρει τα δεδομένα από το ενδιάμεσο αρχείο και επιστρέφεται ως απάντηση στο AJAX που εκτέλεσε το σενάριο. Έτσι το πρόγραμμα JavaScript που θα εκτελέσει τις λειτουργίες του εργαλείου, έχει πλέον τα αποτελέσματα σε μία εύκολα κατανοητή και επεξεργάσιμη μορφή γι' αυτό.

3.4.2 Ανάλυση Τεχνικής Σχεδίασης Κυματομορφών

Η κύρια λειτουργία του εργαλείου σχεδίασης κυματομορφών είναι η εμφάνιση των σημάτων ως κυματομορφές. Η σχεδίαση αυτή γίνεται με την χρήση του στοιχείου 'canvas' που εισάγει η HTML5. Το στοιχείο αυτό μας δίνει την δυνατότητα να σχεδιάσουμε πάνω του γραφικά μέσω ενός προγράμματος JavaScript. Αξίζει να αναφέρουμε ότι παρόμοιες δυνατότητες προσφέρει το στοιχείο SVG το οποίο θα είχαμε προτιμήσει εάν δεν χρειαζόμασταν τις επιπλέον λειτουργίες για την ανάλυση των αποτελεσμάτων. Παρόλα αυτά, καθώς αποσκοπούμε σε ένα αρκετά δυναμικό εργαλείο του οποίου τα στοιχεία που εμφανίζονται μπορεί να υποστούν αρκετές αλλαγές κατά την αλληλεπίδρασή τους με τον χρήστη, καταλήξαμε τελικά στην χρήση του στοιχείου 'canvas'.

Η εμφάνιση των κυματομορφών χωρίζεται σε δύο μέρη. Στο πρώτο μέρος έχουμε τους γενικούς υπολογισμούς και την αρχικοποίηση του στοιχείου 'canvas' αλλά και τον εξωτερικό συντονισμό του δεύτερου μέρους όπου προχωράμε στην σχεδίαση κάθε επιλεγμένου κύματος ξεχωριστά.

```
1. var time_interval=1;
2. var x_multi_intervals=0;
3. var first_timeframe=1;
4. var time_frame = signal_data['time_info']['duration'] / (sig-
   nal_data['time_info']['intervals']-1);
5. var second_divisions = ["ds", "cs", "ms", "µs", "ns", "ps", "fs", "as",
   "zs", "yz"];
6. var second_div_index=second_divisions.indexOf(signal_data['time_in-
   fo']['timescale']);
7. var sub=scale_time_subdivisions(time_frame,0);
8. var second_unit=second_divisions[second_div_index-sub];
9. max_limit = low_limit + Math.round( signal_data['time_info']['inter-
   vals']*($("#simulation_zoom").val()/100) );
10. if(max_limit > signal_data['time_info']['intervals']){
11.   low_limit = low_limit - (max_limit - signal_data['time_info']['inter-
   vals']);
12.   max_limit = signal_data['time_info']['intervals'];
13. }
14.
15. time_interval = max_limit - low_limit;
16. if( time_interval > 9){
17.   time_interval=Math.round(time_interval/9);
18. }else{
19.   time_interval=1;
20. }
```

Κώδικας 31: waveforms_viewer.js - Αρχικοποιήσεις και υπολογισμοί μεταβλητών για την σχεδίαση των κυματομορφών

Οι γενικοί υπολογισμοί συμπεριλαμβάνουν τον καθορισμό της χρονικής υποδιαίρεσης που θα χρησιμοποιήσουμε για την εμφάνιση (Κώδικας 31, γραμμές 4-8), το μικρότερο χρονικό διάστημα μεταξύ πιθανών αλλαγών (Κώδικας 31, γραμμή 4), την ελάχιστη και μέγιστη χρονική στιγμή που θα εμφανίσουμε αναλόγως της χρονικής κλίμακας και μετατόπισης (Κώδικας 31, γραμμές 9-20), το μέγεθος κάθε παλμού σε pixel καθώς και το ύψος που πρέπει να έχει το στοιχείο 'canvas' για την εμφάνιση όλων των σημάτων που έχουν επιλεγεί. Έπειτα αρχικοποιούμε τις μεταβλητές σχεδίασης όπως το χρώμα και γραμματοσειρά που θα χρησιμοποιηθούν, το μέγεθος κενών μεταξύ διάφορων στοιχείων καθώς και το τελικό μέγεθος του στοιχείου 'canvas'. Αφού έχουμε πλέον αρχικοποιήσει και υπολογίσει τα δεδομένα που θα χρειαστούμε, γίνεται η σχεδίαση του χρονικού πλέγματος. Τέλος, προχωράμε στην σχεδίαση κάθε κυματομορφής ξεχωριστά προσθέτοντας τα απαραίτητα στοιχεία για σήματα με μήκος μεγαλύτερο του ένα καθώς δίνουμε στον χρήστη την επιλογή επέκτασής τους.

```
1. for(current_time=0;current_time<=duration;current_time+=time_frame){
2.   if(typeof signal_data[current_time] != 'undefined'){
3.     old_value = signal_data[current_time];
4.     value = signal_data[current_time];
5.
6.   }else{
7.     value=-1;
8.   }
9.   if(!isNaN(current_time) && current_time >= low_limit*time_frame &&
   current_time < max_limit*time_frame){
```

Κώδικας 32: waveforms_viewer.js - Ο κύριος βρόγχος σχεδίασης κυματομορφών και τα αρχικά βήματα προσδιορισμού της τιμής του σήματος

Ο σχεδιασμός κάθε κυματομορφής γίνεται μέσα σε έναν βρόγχο. Ο βρόγχος αυτός ξεκινάει από το μηδέν και φτάνει μέχρι την χρονική διάρκεια προσομοίωσης με βήμα το ελάχιστο χρονικό διάστημα μεταξύ αλλαγών (Κώδικας 32, γραμμή 1). Έτσι περνάμε κάθε φορά από μία χρονική στιγμή προσομοίωσης με πιθανή αλλαγή. Καθώς στα δεδομένα του σήματος συγκρατούμε μόνο τις χρονικές στιγμές στις οποίες υπάρχει αλλαγή, μπορούμε εύκολα να ελέγξουμε σε κάθε επανάληψη εάν υπάρχει νέα τιμή στο σήμα για την συγκεκριμένη χρονική στιγμή (Κώδικας 32, γραμμές 2-8). Καταλήγοντας λοιπόν στην τωρινή τιμή του σήματος επιβεβαιώνουμε ότι ο χρόνος που εξετάζουμε είναι στα χρονικά πλαίσια που θα σχεδιαστούν και προχωρούμε στην σχεδίαση της κυματομορφής (Κώδικας 32, γραμμή 9).

```
1.  if(signal_data['length']==1){
2.    if(value == -1){
3.      value = old_value;
4.    }
5.    // fix the pulse height depending the value
6.    if (value==1){
7.      pulse_height=-10;
8.    }else if(value==0){
9.      pulse_height=10;
10.   }else if(value=='U'){
11.     // unknown value pulse height : 0, color : red
12.     pulse_height =0;
13.     ctx.strokeStyle = "#f00";
14.   }
15.   if(old_pulse_height == -1){
16.     old_pulse_height = pulse_height;
17.   }
18.    // create the horizontal pulse for this time frame
19.    ctx.moveTo(x, y+ pulse_height);
20.    ctx.lineTo(x+x_interval, y+ pulse_height);
21.    // create the vertical pulse for this time frame
22.    ctx.moveTo(x, y+ old_pulse_height);
23.    ctx.lineTo(x, y+ pulse_height);
```

Κώδικας 33: waveforms_viewer.js - Σχεδίαση παλμών για σήματα με μήκος ένα

Η σχεδίαση της κυματομορφής διαχωρίζεται σε σχεδίαση σήματος μήκους ενός ψηφίου και σχεδίαση σήματος με μεγαλύτερο μήκος. Στην πρώτη περίπτωση, παίρνουμε την τωρινή τιμή του σήματος και θέτουμε το ύψος παλμού αντίστοιχα με αυτή (Κώδικας 33, γραμμές 1-17). Στην συνέχεια δημιουργούμε τον οριζόντιο παλμό και την κάθετη γραμμή που τον ενώνει με τον προηγούμενο παλμό (Κώδικας 33, γραμμές 19-23).

Στην δεύτερη περίπτωση όπου έχουμε σήματα με μήκος μεγαλύτερο από ένα ο σχεδιασμός διαφέρει σημαντικά. Αντί για παλμούς το σήμα αποτελείται από δύο οριζόντιες γραμμές που πλαισιώνουν την ονομαστική τιμή του σήματος κατά την διάρκειά τους εάν υπάρχει χώρος για την εμφάνισή του. Έτσι, ελέγχουμε αρχικά εάν υπάρχει αλλαγή τιμής και εάν δεν υπάρχει, υπολογίζουμε τον διαθέσιμο χώρο και σχεδιάζουμε τις οριζόντιες γραμμές (Κώδικας 34, γραμμές 1-3 και 31-38). Όταν υπάρξει αλλαγή τιμής μεταφράζουμε την τιμή στον τύπο που θα εμφανιστεί και μετράμε το απαραίτητο μέγεθος για την εμφάνισή της σε pixel (Κώδικας 34, γραμμές 6-19). Έπειτα, εάν υπάρχει χώρος, εμφανίζουμε την τιμή του σήματος, επαναρχικοποιούμε τον διαθέσιμο χώρο και σχεδιάζουμε μία κάθετη γραμμή με έναν κύκλο στο κέντρο της ώστε να σημειώσουμε οπτικά το σημείο όπου έγινε η αλλαγή (Κώδικας 34, γραμμές 20-28). Αξίζει εδώ να σημειωθεί ότι σε κάθε αλλαγή εμφανίζουμε την παλιά τιμή, πράγμα που μας επιτρέπει τον υπολογισμό του διαθέσιμου χώρου κι έτσι παρατηρούμε ότι προσπερνάμε την πρώτη αλλαγή τιμής (Κώδικας 34, γραμμή 5) ενώ υπάρχει κώδικας στην συνέχεια του αρχείου που αναλαμβάνει την εμφάνιση της τελευταίας τιμής του σήματος που θα σχεδιαστεί.

```
1.  if(value == -1){
2.    value = old_value;
3.    x_multi_intervals = x_interval + x_multi_intervals;
4.  }else{
5.    if(first_timeframe != 1){
6.      data_type=$("#waveform_data_type").val();
7.      if(old_multivalue_named.match(/bU.*/g)){
8.        value_length=old_multivalue_named.length*10;
9.        data_val=old_multivalue_named.substr(1);
10.     }else{
11.       if(data_type=='3'){
12.         data_val="D"+parseInt(old_multivalue_named.sub-
    str(1),2).toString(10);
13.       }else if(data_type=='2'){
14.         data_val="H"+parseInt(old_multivalue_named.sub-
    str(1),2).toString(16);
15.       }else{
16.          data_val="B"+old_multivalue_named.substr(1);
17.       }
18.       value_length=data_val.length*8;
19.     }
20.     if(value_length < x_multi_intervals){
21.       ctx.fillText(data_val,x-x_multi_intervals+5,y+5);
22.     }
23.   }
24.   x_multi_intervals=x_interval;
25.   ctx.moveTo(x, y+10);
26.   ctx.lineTo(x, y-10);
27.   ctx.moveTo(x+1.5, y-1.5);
28.   ctx.arc(x, y, 3, 0, 2 * Math.PI, false);
29. }
30.
31. if( old_value.match(/bU.*/g)){
32.   ctx.strokeStyle = "#f00";
33. }
34.
35. ctx.moveTo(x, y+10);
36. ctx.lineTo(x+x_interval, y+10);
37. ctx.moveTo(x, y-10);
38. ctx.lineTo(x+x_interval, y-10);
```

Κώδικας 34: waveforms_viewer.js - Σχεδίαση παλμών για σήματα με μήκος μεγαλύτερο από ένα

Έτσι ολοκληρώνεται η σχεδίαση κυματομορφών κάθε σήματος μετά την οποία επιστρέφουμε στον εξωτερικό βρόγχο του πρώτου μέρους της σχεδίασης των κυματομορφών για τις απαραίτητες διορθώσεις των τιμών ύψους και πλάτους καθώς και την εισαγωγή στοιχείων για τα σήματα με μήκος μεγαλύτερο του ένα. Ένα ακόμη στοιχείο που αξίζει να αναφέρουμε εδώ είναι ότι όταν ολοκληρωθεί η σχεδίαση όλων των κυματομορφών στο στοιχείο ‘canvas’, αντιγράφουμε τα περιεχόμενα αυτού του στοιχείου σε ένα δεύτερο το οποίο λειτουργεί ως ενδιάμεση μνήμη και επιτρέπει πολλές από τις λειτουργίες που θα περιγράψουμε στο επόμενο υποκεφάλαιο να λειτουργήσουν χωρίς την συνεχή επανασχεδίαση όλων των κυματομορφών.

3.4.3 Ανάλυση Υλοποίησης Βασικών Λειτουργιών

Οι βασικές λειτουργίες που προσφέρονται επιτρέπουν στον χρήστη την ευκολότερη ανάλυση των αποτελεσμάτων προσομοίωσης. Οι λειτουργίες αυτές είναι οδηγούμενες από γεγονότα και ενεργοποιούνται από συγκεκριμένες κινήσεις του χρήστη. Συγκεκριμένα θα περιγράψουμε παρακάτω την υλοποίηση της χρονικής κλίμακας, της χρονικής μετατόπισης, της δημιουργίας προσωρινής και μόνιμης σήμανσης επιπλέον πληροφοριών καθώς και την δυνατότητα λήψης αντιγράφου ως εικόνα.

```
1.  // Slider's event listener
2.  $("#slider_button").mousedown(function(event){
3.    if(window.wave_data){
4.      $('body').on('mousemove', function(e) {
5.        // Calculate the slider's value
6.        offset=e.pageX - $('#slider_main').offset().left  -
    $('.slider_button').outerWidth()/2;
7.        offset = Math.round(offset);
8.        if(offset<0){ offset =1; }
9.        if(offset>100){ offset =100; }
10.       // move the slider and the info box
11.       $("#slider_button").css('left', offset+'px');
12.       // update the time scale value and redraw the data
13.       $("#simulation_zoom").val(offset);
14.       draw_wave(wave_data);
15.       // update the info value of the info box
16.       $("#slider_info").html(offset +'%');
17.       // prevent default behaviour of mouse drag
18.       e.preventDefault();
19.     }).on('mouseup', function() {
20.       $('body').unbind( "mousemove");
21.     });
22.   }
23. });
```

Κώδικας 35: waveforms_viewer.js - Αντιμετώπιση γεγονότος για την λειτουργία χρονικής κλίμακας

Η λειτουργία αλλαγής χρονικής κλίμακας ενεργοποιείται με την χρήση ενός ολισθητή που αντιπροσωπεύει το ποσοστό της κυματομορφής που θα εμφανίζεται. Αρχικά εισάγουμε ένα γεγονός όπου ενεργοποιείται όταν ο χρήστης πατήσει πάνω στο κουμπί του ολισθητή και έπειτα, εάν υπάρχουν δεδομένα σημάτων, αντιμετωπίζουμε την κίνηση του ποντικιού και κατά συνέπεια του ολισθητή (Κώδικας 35, γραμμές 2-4). Στην συνέχεια υπολογίζουμε την θέση του κουμπιού πάνω στον ολισθητή και την περιορίζουμε σε τιμές από 1 έως 100 (Κώδικας 35, γραμμές 6-9). Έπειτα κάνουμε την απαραίτητη διόρθωση στην θέση εμφάνισης του κουμπιού, ανανεώνουμε την τιμή ενός στοιχείου όπου συγκρατεί την τιμή χρονικής κλίμακας, επανασχεδιάζουμε τις κυματομορφές στην νέα κλίμακα, ανανεώνουμε την τιμή που εμφανίζεται δίπλα στον ολισθητή και αποτρέπουμε την προκαθορισμένη ενέργεια του περιηγητή όταν σύρεται το ποντίκι (Κώδικας 35, γραμμές 11-18). Έτσι ολοκληρώνεται η πρώτη λειτουργία που επιτρέπει την σμίκρυνση ή μεγέθυνση των κυματομορφών και δίνει στον χρήστη την δυνατότητα εξέτασης των σημάτων με μεγαλύτερη ακρίβεια.

```
1. $("#WavesCanvas").mousedown(function(event){
2.   [...]
3.   isDown = true;
4.   delay_pos=0;
5.   delay_neg=0;
6.   mouse_down_x= event.pageX;
7. });
8.
9. $( "#WavesCanvas" ).mousemove(function( event ) {
10.   if(isDown){
11.     var mouse_x= event.pageX;
12.     var dif = mouse_down_x-mouse_x;
13.     if(dif>0){
14.       delay_pos = dif/10 + delay_pos;
15.       if(delay_pos>=1){
16.         delay_pos=0;
17.         dif=1;
18.       }else{
19.         dif=0;
20.       }
21.     }
22.     if(dif<0){
23.       delay_neg = dif/10 + delay_neg;
24.       if(delay_neg<=-1){
25.         delay_neg=0;
26.         dif=-1;
27.       }else{
28.         dif=0;
29.       }
30.     }
```

Κώδικας 36: waveforms_viewer.js - Αρχικοποίηση μεταβλητών στο πάτημα ποντικιού και η λειτουργία ομαλοποίησης της χρονικής μετατόπισης

Η χρονική μετατόπιση είναι μία λειτουργία που επιτρέπει μετά την αλλαγή της χρονικής κλίμακας σε ποσοστό μικρότερο του 100% την ολίσθηση των κυματομορφών κατά τον χρονικό άξονα. Η λειτουργία αυτή ενεργοποιείται όταν ο χρήστης πατήσει και σύρει είτε το δεξιό ή το αριστερό κουμπί του ποντικιού πάνω στο στοιχείο ‘canvas’. Με το πάτημα το ποντικιού αρχικοποιούμε κάποιες μεταβλητές (Κώδικας 36, γραμμές 1-7) τις οποίες χρησιμοποιούμε κατά την μετακίνηση του. Έτσι κατά την μετακίνηση του ποντικιού πάνω στο στοιχείο ‘canvas’ επιβεβαιώνουμε ότι το κουμπί είναι πατημένο και υπολογίζουμε την μετατόπιση του δείκτη (Κώδικας 36, γραμμές 9-12). Εάν η μετατόπιση είναι θετική, ο δείκτης κινείται προς τα αριστερά της οθόνης ενώ εάν είναι αρνητική ο δείκτης κινείται προς τα δεξιά. Σε κάθε περίπτωση χρησιμοποιούμε μία μεταβλητή καθυστέρησης ώστε η ολίσθηση να είναι αρκετά ομαλή. Συγκεκριμένα παρατηρήσαμε ότι η μετατόπιση του χρόνου κατά ένα ελάχιστο χρονικό διάστημα πιθανής αλλαγής τιμής σημάτων στα δέκα pixel μετατόπισης του δείκτη ποντικιού (Κώδικας 36, γραμμές 14-20 και 23-29) επιτρέπει μία αρκετά ομαλή ολίσθηση που παρέχει επαρκώς την αίσθηση ανατροφοδότησης από την κίνηση του χρήστη.

```
1.  mouse_down_x = mouse_x;
2.  temp_low_limit = low_limit +dif;
3.  temp_max_limit = max_limit +dif;
4.  if(temp_low_limit < 0){
5.    low_limit=0;
6.  }else{
7.    if(temp_max_limit > wave_intervals){
8.      low_limit = wave_intervals - (max_limit - low_limit);
9.      max_limit = wave_intervals;
10.   }else{
11.     low_limit = temp_low_limit;
12.     max_limit = temp_max_limit;
13.   }
14. }
15. if(dif!=0){
16.   draw_wave(wave_data);
17. }
```

Κώδικας 37: waveforms_viewer.js - Υπολογισμός νέων ορίων προβολής αποτελεσμάτων και επανασχεδίαση εάν υπάρχει χρονική μετατόπιση

Συνεχίζοντας, υπολογίζουμε τα νέα όρια κυματομορφών που θα προβάλλονται κατά την σχεδίασή τους και επιβεβαιώνουμε ότι δεν ξεπερνούν τα όρια προσομοίωσης (Κώδικας 37, γραμμές 2-14). Τέλος, εάν υπάρχει επαρκή μετακίνηση του δείκτη ποντικιού για την ολίσθηση, επανασχεδιάζουμε τις κυματομορφές (Κώδικας 37, γραμμές 15-17). Έτσι ολοκληρώνεται η λειτουργία χρονικής μετατόπισης που επιτρέπει στον χρήστη την ολίσθηση κατά τον χρονικό άξονα των κυματομορφών ενώ παραμένει σε εστιασμένη χρονική κλίμακα.

Οι σηματοδότες επιπλέον πληροφοριών έχουν παρόμοια υλοποίηση με κάποιες μικρές διαφορές. Ο προσωρινός σηματοδότης καλείται στο γεγονός κίνησης του δείκτη ποντικιού πάνω από τις κυματομορφές ενώ ο μόνιμος στο γεγονός πατήματος ενός κουμπιού από το ποντίκι. Επίσης ο μόνιμος σηματοδότης μεταφέρεται και στο δεύτερο στοιχείο ‘canvas’ που λειτουργεί ως προσωρινή μνήμη ενώ ο προσωρινός μένει μόνο στο εμφανές στοιχείο ‘canvas’ με αποτέλεσμα να απομακρύνεται κατά την μετακίνηση του ποντικιού.

```
1. $("#WavesCanvas").mousedown(function(event){
2.   x=event.pageX - $('#WavesCanvas').offset().left;
3.   canvas_width = $('#WavesCanvas').width();
4.   if(x > 110 && x < (canvas_width-20)){
5.     draw_wave(wave_data);
6.     var second_divisions = ["ds", "cs", "ms", "μs", "ns", "ps", "fs",
   "as", "zs", "yz"];
7.     var second_div_index=second_divisions.indexOf(wave_data['time_in-
   fo']['timescale']);
8.     x_interval =(canvas_width-130) / (max_limit - low_limit) ;
9.     time_interval = wave_data['time_info']['duration'] / (
   wave_data['time_info']['intervals'] -1);
10.     timeframe =( ( Math.floor((x-110) / x_interval) + low_limit ) *
   time_interval );
11.     sub=scale_time_subdivisions(time_interval,0);
12.     timeframe_str = timeframe / (Math.pow(1000, sub));
```

Κώδικας 38: waveforms_viewer.js - Γεγονός πατήματος ποντικιού, αρχικοποίηση και υπολογισμός μεταβλητών για τον μόνιμο σηματοδότη επιπλέων πληροφοριών

Τέλος, για λόγους ευκρίνειας και κατανόησης των λειτουργιών από τον χρήστη, τα δύο διαφορετικά είδη σηματοδότη χρησιμοποιούν διαφορετικά χρώματα όπου ο προσωρινός είναι κόκκινος και ο μόνιμος είναι μπλε. Στην συνέχεια θα εξετάσουμε την υλοποίηση του μόνιμου σηματοδότη. Αρχικά στην εισαγωγή του μόνιμου σηματοδότη έχουμε την επανασχεδίαση των κυματομορφών για την απομάκρυνση προηγούμενων σηματοδοτών (Κώδικας 38, γραμμή 5). Η ενέργεια αυτή στον προσωρινό σηματοδότη γίνεται με αντικατάσταση των στοιχείων του 'canvas' με αυτά του δεύτερου 'canvas' που λειτουργεί ως προσωρινή μνήμη. Έπειτα έχουμε το υπολογισμό της χρονικής υποδιαίρεσης που θα χρησιμοποιήσουμε καθώς και του χρόνου πάνω στον οποίο βρίσκεται ο δείκτης ποντικιού (Κώδικας 38, γραμμές 6-12). Στην συνέχεια αρχικοποιούμε κάποιες μεταβλητές για την σχεδίαση όπως τα χρώματα, γραμματοσειρές και αποστάσεις.

```
1. $.each(window.selected_signals, function( i, index ) {
2.   for(current_time=0;current_time<=timeframe;current_time+=time_inter-
   val){
3.     if(typeof wave_data[index][current_time] != 'undefined'){
4.       wave_val = wave_data[index][current_time];
5.       if(wave_data[index]['length']>1){
6.         wave_val_bin = wave_val;
7.         data_type=$("#waveform_data_type").val();
8.         if(wave_val.match(/bU.*/g)){
9.           wave_val=wave_val.substr(1);
10.        }else{
11.          if(data_type=='3'){
12.            wave_val="D"+parseInt(wave_val.substr(1),2).toString(10);
13.          }else if(data_type=='2'){
14.            wave_val="H"+parseInt(wave_val.substr(1),2).toString(16);
15.          }else{
16.            wave_val="B"+wave_val.substr(1);
17.          }
18.        }
19.      }
20.    }
21.  }
```

Κώδικας 39: waveforms_viewer.js - Εύρεση τιμής για κάθε σήμα για τον μόνιμο σηματοδότη επιπλέων πληροφοριών

Έπειτα βρίσκουμε την τιμή του κάθε επιλεγμένου σήματος για την χρονική στιγμή που υπολογίσαμε παραπάνω. Για να το επιτύχουμε αυτό ελέγχουμε τις αλλαγές τιμής από τον χρόνο μηδέν έως το χρόνο που μας ενδιαφέρει και κρατάμε την τελευταία τιμή που βρέθηκε ως την τιμή του σήματος (Κώδικας 39). Αυτή η τεχνική γίνεται απαραίτητη καθώς συγκρατούμε μόνο τις αλλαγές τιμών και είναι πολύ πιθανό η συγκεκριμένη χρονική στιγμή να μην περιέχει αλλαγή για κάθε σήμα. Κατά την αναζήτηση αυτή γίνεται και η μετάφραση των τιμών στον τύπο που έχει επιλέξει ο χρήστης για τα σήματα με μήκος μεγαλύτερο από ένα.

```
1.  ctx.fillText("Value : "+wave_val,x_canvas,y_canvas);
2.  y_canvas=y_canvas+40;
3.  if(wave_data[index]["expand"]==1){
4.    for(j=1;j<=wave_data[index]["length"];j++){
5.      sub_wave_val=wave_val_bin.charAt(j);
6.      ctx.fillText("Value : "+sub_wave_val,x_canvas+15,y_canvas);
7.      y_canvas=y_canvas+40;
8.    }
9.  }
```

Κώδικας 40: waveforms_viewer.js - Εμφάνιση τιμής σήματος στον μόνιμο σηματοδότη επιπλέον πληροφοριών

Στην συνέχεια εμφανίζουμε την τιμή κάθε σήματος κάτω από την ονομασία του καλύπτοντας την περίπτωση σημάτων με μήκος μεγαλύτερο από ένα που έχουν επεκταθεί όπου προσάπτουμε την τιμή κάθε υποσήματος κάτω από το όνομά του. Τέλος σχεδιάζουμε μία κάθετη γραμμή κατά ύψος του στοιχείου ‘canvas’ στο σημείο όπου βρίσκεται ο δείκτης ποντικιού και σημειώνουμε από κάτω τον χρόνο που αντιστοιχεί σε αυτόν τον σηματοδότη. Έτσι υλοποιούμε μία από τις σημαντικότερες λειτουργίες του εργαλείου που δίνει στον χρήστη την δυνατότητα να πάρει άμεσα και εύκολα πληροφορίες για την τιμή κάθε σήματος σε κάθε χρονική στιγμή που εμφανίζεται.

Υπάρχουν μερικές ακόμη λειτουργίες που έχουμε ήδη εξετάσει μέσα από άλλες. Για παράδειγμα η επιλογή τύπου εμφάνισης των τιμών όπου υποστηρίζονται ο δυαδικός, δεκαδικός και δεκαεξαδικός τύπος. Η υλοποίηση αυτής της λειτουργίας έγινε εμφανής κατά την εμφάνιση των τιμών στα σήματα με μήκος μεγαλύτερο του ένα. Μία άλλη λειτουργία που εξετάστηκε επιφανειακά είναι η επέκταση σημάτων με μεγάλο μήκος. Η υλοποίηση αυτής της λειτουργίας γίνεται με την εισαγωγή στοιχείων που έχουν ξεχωριστά γεγονότα. Όταν ένα από αυτά τα γεγονότα ενεργοποιηθεί το αντίστοιχο σήμα σημειώνεται ως επεκταμένο ή συμπυκνωμένο και επανασχεδιάζουμε τις κυματομορφές. Στην σχεδίαση των κυματομορφών, τα σήματα που έχουν επεκταθεί εισάγουν ένα νέο σήμα μήκους ενός ψηφίου για κάθε ξεχωριστό δυαδικό ψηφίο τους και ομαδοποιούνται κάτω από το σήμα μεγάλου μήκους που τα περιέχει.

```
1.  $("#btn-download-image").click(function(event){
2.    var canvas = document.getElementById("WavesCanvas");
3.    var dataURL = canvas.toDataURL('image/png');
4.    var img = canvas.toDataURL("image/jpg");
5.    $("#btn-download-image").attr('href', dataURL);
6.  });
```

Κώδικας 41: waveforms_viewer.js - Δημιουργία και μεταφόρτωση εικόνας από το στοιχείο 'canvas'

Τέλος έχουμε την λειτουργία λήψης στιγμιότυπου ως εικόνα. Η λειτουργία αυτή ενεργοποιείται από το γεγονός κατά το οποίο ο χρήστης πατάει το αντίστοιχο κουμπί και δημιουργεί μία εικόνα τύπου jpg από το στοιχείο ‘canvas’ που μεταφορτώνεται στον υπολογιστή του χρήστη (Κώδικας 41).

Ολοκληρώνουμε έτσι την ανάλυση της υλοποίησης του εργαλείου σχεδίασης κυματομορφών. Γίνεται πλέον κατανοητό ότι το εργαλείο αυτό δεν αποσκοπεί μόνο στην προβολή των αποτελεσμάτων μίας προσομοίωσης. Παρέχει επί πλέον πολλαπλά στοιχεία και εισάγει λειτουργείες που προσθέτουν την δυνατότητα διερεύνησης και προσαρμογής των κυματομορφών για πιο άμεση και εύκολη ανάλυση τους από τον χρήστη.

3.5 Οδηγίες Εγκατάστασης

Η εφαρμογή που αναπτύξαμε είναι μία εφαρμογή διαδικτύου κι έτσι απαιτεί την διατήρηση ενός εξυπηρετητή για την ορθή λειτουργία της. Αρχικά, ο εξυπηρετητής θα πρέπει να χρησιμοποιεί ένα λειτουργικό σύστημα UNIX το οποίο είναι σε θέση να υποστηρίξει τις απαραίτητες κλήσεις συστήματος και την εκτέλεση των σεναρίων τερματικού που χρησιμοποιούμε. Τα περισσότερα ολοκληρωμένα λειτουργικά συστήματα LINUX μπορούν εύκολα να καλύψουν αυτές τις προϋποθέσεις.

```
1. wget http://ghdl.free.fr/ghdl-0.29-i686-pc-linux.tar
2. tar -xvf ghdl-0.29-i686-pc-linux.tar
3. cd ghdl-0.29-i686-pc-linux
4. sudo tar -C / -jxvf ghdl-0.29-i686-pc-linux.tar.bz2
```

Κώδικας 42: Εντολές εγκατάστασης του προγράμματος GHDL

Καθώς η εφαρμογής μας βασίζεται στο εξωτερικό πρόγραμμα GHDL θα πρέπει να βεβαιωθούμε ότι υπάρχει και στον εξυπηρετητή μας. Ο έλεγχος μπορεί εύκολα να γίνει ανοίγοντας ένα τερματικό και δίνοντας την εντολή ‘ghdl’. Εάν μας επιστραφεί μήνυμα ότι η εντολή δεν αναγνωρίζεται, τότε το πρόγραμμα δεν έχει εγκατασταθεί. Η εγκατάσταση της τελευταίας έκδοσης του GHDL γίνεται όπως φαίνεται παραπάνω από τις 4 διαδοχικές εντολές σε ένα τερματικό (Κώδικας 42).

```
1. sudo apt-get install mysql-server
2. sudo mysql_secure_installation

3. sudo apt-get install php-fpm php-mysql
```

Κώδικας 43: Εντολές εγκατάστασης της MySQL και PHP

Αφού έχουμε εγκαταστήσει το GHDL μπορούμε να προχωρήσουμε στην εγκατάσταση των βασικών προγραμμάτων κάθε εξυπηρετητή. Ξεκινώντας με την βάση δεδομένων MySQL η εγκατάσταση γίνεται με την εντολή (Κώδικας 43, γραμμή 1) κατά την διάρκεια της οποίας θα ζητηθεί να εισάγουμε έναν κωδικό για την πρόσβαση του χρήστη ‘root’. Έπειτα μπορούμε να εκτελέσουμε την εντολή (Κώδικας 43, γραμμή 2) με την οποία θα περάσουμε κάποια βήματα εισαγωγής μεγαλύτερης ασφάλειας στην βάση δεδομένων. Παρόλο που συνιστάται, η δεύτερη εντολή δεν είναι αναγκαία. Στην συνέχεια μπορούμε να προχωρήσουμε στην εγκατάσταση της PHP με την εντολή (Κώδικας 43, γραμμή 3).

Σε αυτό το σημείο θα καλύψουμε την εγκατάσταση του εξυπηρετητή HTTP. Καθώς υπάρχουν διάφορες επιλογές στα προγράμματα που μπορούν να καλύψουν αυτή την ανάγκη, θα δώσουμε οδηγίες για δύο από τα δημοφιλέστερα στον τομέα, το Apache και το nginx.

```
1. sudo apt-get install apache2
2. sudo nano /etc/apache2/apache2.conf
3. sudo service apache2 restart
```

Κώδικας 44: Εντολή εγκατάστασης, επεξεργασίας των ρυθμίσεων και επανεκκίνησης του Apache

Η εγκατάσταση του HTTP εξυπηρετητή Apache γίνεται με την εντολή (Κώδικας 44, γραμμή 1). Μετά την ολοκλήρωση της εγκατάστασης, μπορούμε να επεξεργαστούμε τις ρυθμίσεις του με την εντολή (Κώδικας 44, γραμμή 2) και να επανεκκινήσουμε την λειτουργία του με την εντολή (Κώδικας 44, γραμμή 3).

```
1. sudo apt-get install nginx
2. sudo gedit /etc/nginx/sites-available/default
3. sudo service nginx start
```

Κώδικας 45: Εντολή εγκατάστασης, επεξεργασίας των ρυθμίσεων και εκκίνησης του nginx

Παρόμοια, η εγκατάσταση του nginx γίνεται με την εντολή (Κώδικας 45, γραμμή 1). Η επεξεργασία των ρυθμίσεων με την εντολή (Κώδικας 45, γραμμή 2) και η εκκίνηση της λειτουργίας του με την εντολή (Κώδικας 45, γραμμή 3). Έτσι ολοκληρώνουμε την εγκατάσταση των απαιτούμενων προγραμμάτων.

Στην συνέχεια είναι απαραίτητη η ρύθμιση αυτών των προγραμμάτων και η τοποθέτηση των αρχείων της εφαρμογής στους κατάλληλους καταλόγους.

```
1. sudo mysql
2.   CREATE DATABASE vhdl_compiler;
3.   USE vhdl_compiler;
4.   CREATE USER '<USERNAME>'@'localhost' IDENTIFIED BY '<PASSWORD>';
5.   GRANT ALL PRIVILEGES ON vhdl_compiler. * TO '<USERNAME>'@'localhost';
6.   EXIT;
7. sudo mysql -u root -p vhd_compiler < /var/www/html/hdl/vhdl_com-
   piler.sql
```

Κώδικας 46: Εντολές ρύθμισης της βάσης δεδομένων

Η ρύθμιση της βάσης δεδομένων γίνεται αρχικά ανοίγοντας σε ένα τερματικό το κέλυφος mysql με την εντολή (Κώδικας 46, γραμμή 1). Εντός του θα πρέπει αρχικά να δημιουργήσουμε την βάση δεδομένων και να την θέσουμε ως την βάση που χρησιμοποιούμε (Κώδικας 46, γραμμές 2-3). Έπειτα δημιουργούμε έναν χρήστη δίνοντας όνομα και κωδικό και του χορηγούμε όλα τα δικαιώματα στην βάση δεδομένων μας (Κώδικας 46, γραμμές 4-5). Τέλος μπορούμε σε αυτό το σημείο να ανοίξουμε το αρχείο vhdl_compiler.sql και να αντιγράψουμε και επικολλήσουμε στο τερματικό τις εντολές του ή να βγούμε από το κέλυφος mysql και να δώσουμε την εντολή για αυτόματη εισαγωγή του αρχείου στην βάση που δημιουργήσαμε (Κώδικας 46, γραμμές 6-7). Τέλος, μπορούμε πλέον να απομακρύνουμε το αρχείο vhdl_compiler.sql από τον αρχικό κατάλογο των αρχείων της εφαρμογής. Στην περίπτωση που ο HTTP εξυπηρετητής που χρησιμοποιούμε είναι ο nginx, θα χρειαστεί μία επιπλέον ρύθμιση για την σωστή λειτουργία της εφαρμογής. Αρχικά το αρχείο .htaccess μπορεί να διαγραφεί από τον αρχικό κατάλογο των αρχείων της εφαρμογής, αφού δεν χρησιμοποιείται. Έπειτα ανοίγουμε το αρχείο ρυθμίσεων του nginx όπως δείξαμε παραπάνω και μεταφέρουμε τα περιεχόμενα του αρχείου nginx_conf.txt στο τέλος των ρυθμίσεων ‘server{}’ αφού διορθώσουμε όπου χρειάζεται τον υπολοκατάλογο στον οποίο θα περασθούν τα αρχεία της εφαρμογής. Κάνουμε επανεκκίνηση της υπηρεσίας nginx και απομακρύνουμε το αρχείο nginx_conf.txt από τον αρχικό κατάλογο των αρχείων της εφαρμογής. Αντίστοιχα για την περίπτωση που χρησιμοποιούμε Apache, θα χρειαστεί πιθανώς να διορθώσουμε τον υποκατάλογο στον οποίο θα μεταφέρουμε τα αρχεία της εφαρμογής στο αρχείο .htaccess. Και

στις δύο περιπτώσεις σαν προκαθορισμένη τιμή θεωρούμε ότι η εφαρμογή θα βρίσκεται από στον υποκατάλογο ‘vhdl’ οποίος θα είναι στον ριζικό κατάλογο όπου ο HTTP εξυπηρετητής προσφέρει στο διαδίκτυο. Τέλος μένει η μεταφορά των αρχείων της εφαρμογής στον επιθυμητό κατάλογο για την διάθεσή τους στο διαδίκτυο και η ρύθμιση κάποιων μεταβλητών τους. Αρχικά, το αρχείο /classes/Database.php περιέχει στις γραμμές 10-13 τα στοιχεία της βάσης δεδομένων. Εκεί θα πρέπει να δοθούν οι τέσσερεις μεταβλητές που περιέχουν το όνομα του εξυπηρετητή (συνήθως μένει ως ‘localhost’), το όνομα του χρήστη της βάσης δεδομένων, τον κωδικό αυτού του χρήστη καθώς και το όνομα της βάσης (που ακολουθώντας αυτές τις οδηγίες θα πρέπει να παραμένει ‘vhdl_compiler’). Στην συνέχεια μπορούν επιλεκτικά να ενημερωθούν τα αρχεία loader.php (γραμμές 17-23) και job_scheduler.sh (γραμμές 20-23) εάν οι επιθυμητοί κατάλογοι για τα προσωρινά αλλά και μόνιμα αρχεία της εφαρμογής είναι διαφορετικοί από τους προεπιλεγμένους. Γίνεται φανερό με την ολοκλήρωση της διαδικασίας ότι η εγκατάσταση της εφαρμογής μας απαιτεί μία εξοικείωση με την χρήση εξυπηρετητών αλλά προσφέρει ταυτόχρονα πολλές επιλογές για την προσαρμογή της στις ανάγκες του υποστηρίζοντας εναλλακτικές λύσεις για τα επιμέρους στοιχεία του. Έτσι ολοκληρώνεται η εγκατάσταση και ρύθμιση της εφαρμογής. Μένει η επανεκκίνηση του εξυπηρετητή εάν δεν έχει γίνει μετά την αλλαγή των ρυθμίσεών του ώστε η εφαρμογή να είναι πλέον λειτουργική.

3.6 Σύνοψη Κεφαλαίου

Σε αυτό το κεφάλαιο αναλύσαμε τον τρόπο σχεδίασης και ανάπτυξης της εφαρμογής. Πρέπει πλέον να είναι κατανοητές οι τεχνικές και η λογική που χρησιμοποιήσαμε για την δημιουργία των επιμέρους λειτουργιών. Αναλύσαμε την εφαρμογή από την οργάνωση των καταλόγων που χρησιμοποιούνται και την σχεδίαση της βάσης δεδομένων ως την ανάπτυξη του κώδικα που αποτελεί την εφαρμογή μας. Τέλος, δόθηκε η δυνατότητα στον αναγνώστη να αξιοποιήσει την εφαρμογή σε έναν εξυπηρετητή ακολουθώντας τις οδηγίες εγκατάστασης. Στο επόμενο κεφάλαιο θα περιγράψουμε τις λειτουργίες που προσφέρει η εφαρμογή μας, έχοντας κατανοήσει τον τρόπο ανάπτυξης αυτών, με την δυνατότητα να τις παρακολουθήσουμε από τον δικό μας εξυπηρετητή τόσο σε επίπεδο πελάτη όσο και διακομιστή.

Κεφάλαιο 4 - Περιγραφή Λειτουργιών

Στο κεφάλαιο αυτό θα περιγράψουμε τις ξεχωριστές λειτουργίες που προσφέρονται καθώς και κάποιες περιπτώσεις χρήσης για την καλύτερη κατανόηση τους. Έχοντας υπόψη τον τρόπο που υλοποιούνται αυτές οι λειτουργίες μπορούμε πλέον να ολοκληρώσουμε την ανάλυσή τους από την οπτική γωνία του χρήστη. Έτσι θα δούμε αρχικά τις λειτουργίες που προσφέρονται στα έργα κάθε χρήστη καθώς και τα περιεχόμενα αυτών, όπως τα αρχεία τους αλλά και τα αποτελέσματα προσομοίωσης. Στην συνέχεια θα δούμε τον τρόπο που εφαρμόζουμε τις λειτουργίες εξαρτημάτων και βιβλιοθηκών από την ανάρτηση έως τον έλεγχο και ενημέρωση αυτών καθώς και τις βοηθητικές λειτουργίες που έχουμε προσάψει στο σύστημά τους όπως η ταξινόμηση, προβολή και αναζήτηση εξαρτημάτων. Τέλος, θα περάσουμε στα επίπεδα χρηστών, έχοντας κατανοήσει τις λειτουργίες που προσφέρει η εφαρμογή μας, όπου θα αναλύσουμε τις διαφορές τους στα επίπεδα πρόσβασης και εξουσιοδότησης που προσφέρουν. Έτσι στο τέλος του κεφαλαίου ο αναγνώστης θα έχει μία πλήρη εικόνα για τον τρόπο που επιτυγχάνουμε τόσο τον σκοπό της εφαρμογής όσο και τους στόχους που θέσαμε.

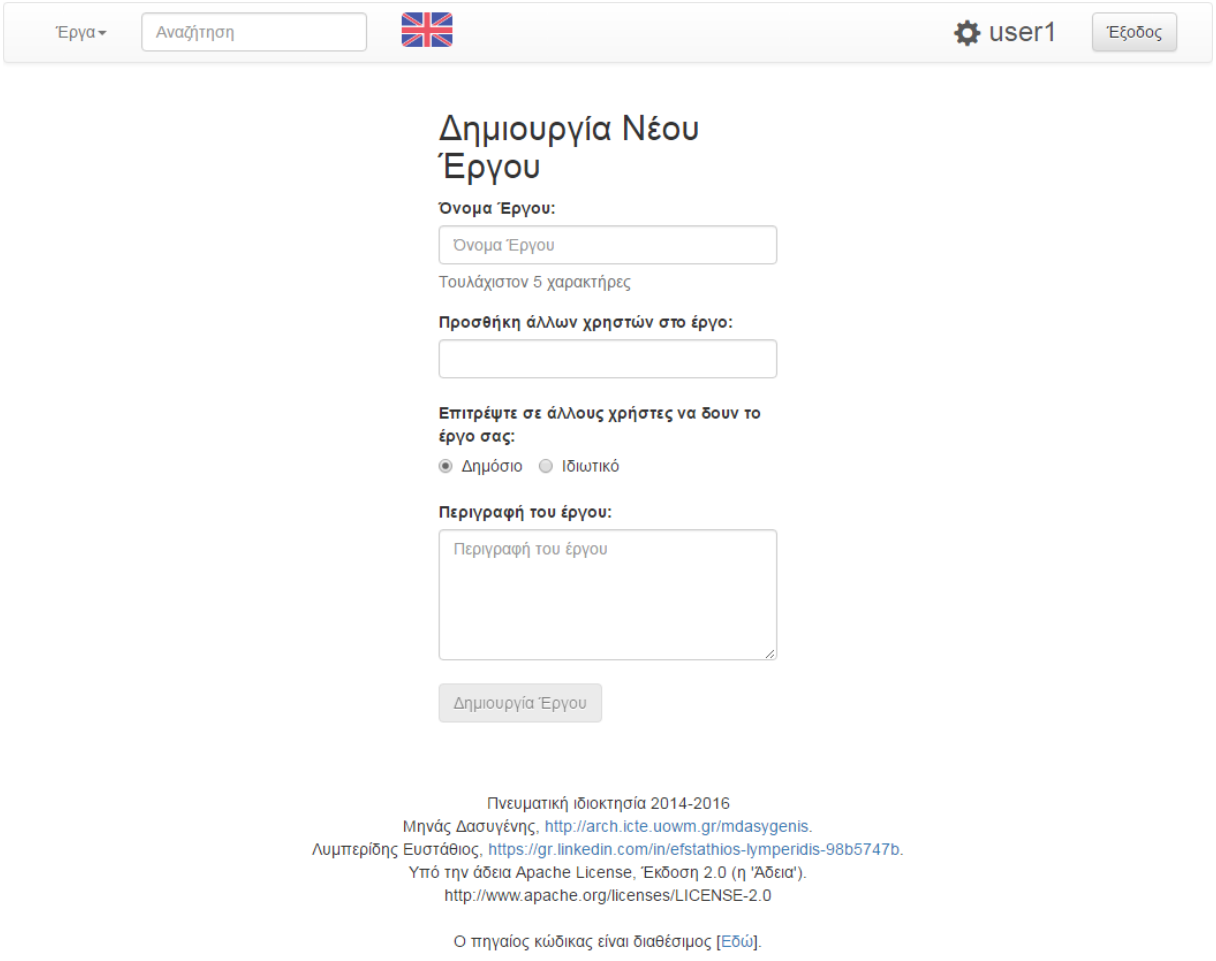
4.1 Λειτουργίες Έργων

Τα έργα χρηστών αποτελούν την βασική λειτουργία της εφαρμογής μας. Ένα έργο αντιπροσωπεύει συνήθως ένα πρόγραμμα και αντιμετωπίζεται ως τέτοιο από τις λειτουργίες που προσφέρουμε. Δεν αποσκοπούμε όμως στον περιορισμό της χρήσης του. Έτσι ένας χρήστης μπορεί να χρησιμοποιήσει ένα έργο ως μία ποιο αυθαίρετη έννοια. Για παράδειγμα

ένας χρήστης μπορεί να διατηρεί ένα έργο για την συλλογή χρήσιμων εξαρτημάτων. Ο μόνος περιορισμός που επιβάλλεται βρίσκεται στο συνολικό μέγεθος των αρχείων που περιλαμβάνει ένα έργο. Έχουμε θέσει τον περιορισμό αυτό στα 20 MegaBytes καθώς το μέγεθος αυτό μπορεί εύκολα να υπερκαλύψει τις ανάγκες ενός VHDL προγράμματος με τα επιπλέον αρχεία μεταγλώττισης και προσομοίωσης. Ταυτόχρονα όμως δεν επιτρέπει υπερβολές σε χρόνους εκτέλεσης και αριθμό εξαρτημάτων. Θα αναλύσουμε παρακάτω τις λειτουργίες που προσφέρουμε σε κάθε έργο έχοντας υπόψη την αναμενόμενη χρήση του. Αρχικά θα δούμε τις λειτουργίες που αφορούν την δημιουργία, διαγραφή και επεξεργασία του έργου ενώ μετά θα προχωρήσουμε στις αντίστοιχες λειτουργίες για τα αρχεία που περιέχει. Έπειτα θα μεταβούμε στις λειτουργίες μεταγλώττισης και προσομοίωσης, τις λειτουργίες ανάλυσης των αποτελεσμάτων προσομοίωσης και τέλος στις δυνατότητες διαμοίρασης των έργων τόσο για την συλλογική ανάπτυξη όσο και την απλή επίδειξή τους.

4.1.1 Δημιουργία, Διαγραφή και Επεξεργασία Έργου

Για την εισαγωγή στις λειτουργίες που προσφέρονται σε κάθε έργο θα δούμε τις λειτουργίες που αποσκοπούν στην διατήρηση του ίδιου του έργου. Οι λειτουργίες αυτές είναι η δημιουργία, διαγραφή και επεξεργασία του έργου.



Εικόνα 11: Οθόνη δημιουργίας νέου έργου

Η δημιουργία ενός έργου γίνεται από την αρχική σελίδα του χρήστη και έχει ως σκοπό την εισαγωγή των στοιχείων του έργου στην βάση δεδομένων αλλά και την δημιουργία των καταλόγων του έργου στον καταλόγου αυτού του χρήστη. Κατά την διαδικασία αυτή ο χρήστης συμπληρώνει τα απαραίτητα στοιχεία σε μία φόρμα ώστε να δημιουργήσει ένα έργο εύκολα αναγνωρίσιμο και με τα χαρακτηριστικά που επιθυμεί όπως φαίνεται στην οθόνη δημιουργίας νέου έργου στην Εικόνα 11. Η διαγραφή του έργου βρίσκεται ως επιλογή στην σελίδα του έργου και είναι προσβάσιμη μόνο από τους χρήστες με δικαιώματα ιδιοκτήτη σε αυτό. Σκοπό της είναι η απομάκρυνση των εγγραφών του έργου από την βάση δεδομένων και η διαγραφή των αρχείων και καταλόγου που αντιστοιχούν σε αυτό. Παρόμοια, η επεξεργασία ενός έργου βρίσκεται στην σελίδα του έργου και είναι προσβάσιμη μόνο από τους χρήστες με δικαιώματα ιδιοκτήτη σε αυτό. Η επεξεργασία έχει πολλά κοινά στοιχεία με την δημιουργία ενός έργου, επιτρέποντας την αλλαγή των στοιχείων που δόθηκαν στην διάρκειά της, από το όνομα του έργου ως τους χρήστες που έχουν πρόσβαση σε αυτό και την περιγραφή του. Καθώς λοιπόν η διαγραφή του έργου έχει ένα βήμα και η επεξεργασία έχει σχεδόν πανομοιότυπη περίπτωση χρήσης με την δημιουργία, παρακάτω παρουσιάζουμε την περίπτωση χρήσης της δημιουργίας ενός έργου.

Περίπτωση Χρήσης: **Δημιουργία Έργου**
Πρωταγωνιστής: Χρήστης
Προϋπόθεση: Ο χρήστης είναι συνδεμένος και βρίσκεται στην αρχική του σελίδα.
Εκπλήρωση: Δημιουργείται το έργο με τα στοιχεία που έδωσε ο χρήστης.
Βασικό Σενάριο:

1. Ο χρήστης πλοηγείται στην επιλογή ‘Δημιουργία Νέου Έργου’.
2. Ο χρήστης μεταφέρεται στην σελίδα δημιουργίας νέου έργου.
3. Ο χρήστης πληκτρολογεί το όνομα του έργου.
4. Ο χρήστης πληκτρολογεί τα ονόματα χρηστών που θέλει να έχουν δικαιώματα επεξεργασίας στο έργο του και επιλέγει τους χρήστες από την λίστα που εμφανίζεται.
5. Ο χρήστης επιλέγει εάν το έργο θα είναι διαθέσιμο στο ευρύ κοινό ή όχι.
6. Ο χρήστης πληκτρολογεί μία περιγραφή του έργου.
7. Ο χρήστης αποστέλλει την φόρμα δημιουργίας επιλέγοντας ‘Δημιουργία Έργου’.
8. Το έργο δημιουργείται και ο χρήστης μεταφέρεται στην σελίδα του συγκεκριμένου έργου.

Εναλλακτικά σενάρια:

3α. Το όνομα έργου είναι μικρότερο από πέντε (5) χαρακτήρες.

3α1. Εμφανίζεται σχετικό μήνυμα λάθους κάτω από το πεδίο κειμένου.

3β. Το όνομα του έργου δεν είναι έγκυρο όνομα καταλόγου.

3β1. Εξάγεται ένα έγκυρο όνομα καταλόγου από το όνομα που δόθηκε και χρησιμοποιείται αυτό στον κατάλογο ενώ εμφανίζεται αυτό που έδωσε ο χρήστης στην σελίδα.

4α. Δεν υπάρχει χρήστης με το όνομα που πληκτρολογήθηκε.

4α1. Η λίστα με τους πιθανούς χρήστες είναι κενή και δεν εμφανίζεται ώστε να επιλεγεί κάποιος χρήστης.

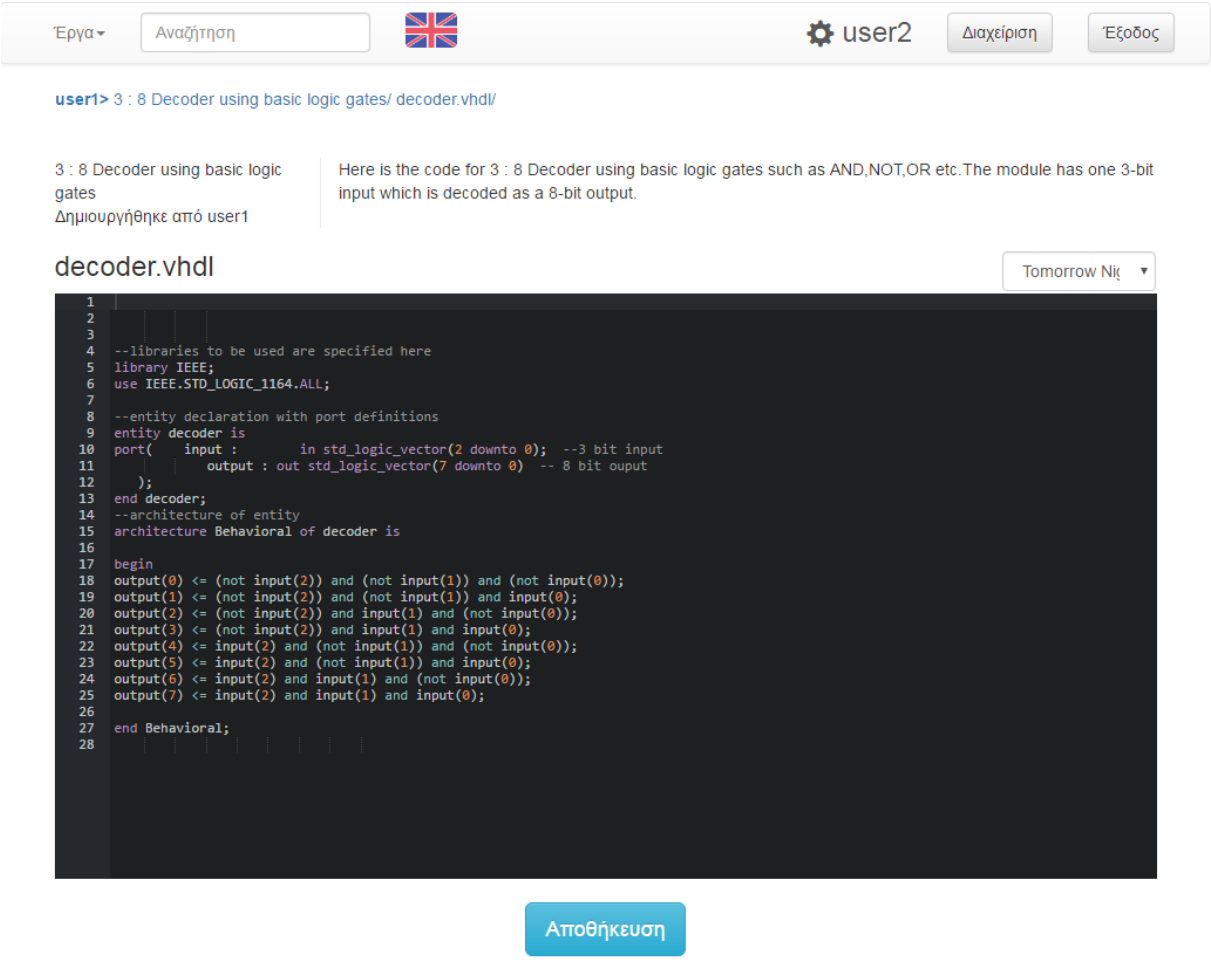
7α. Ο χρήστης δεν έχει συμπληρώσει όλα τα απαραίτητα πεδία.

7α1. Το κουμπί αποστολής φόρμας εμφανίζεται με γκρι απόχρωση και δεν εκτελεί την λειτουργία αποστολής.

4.1.2 Δημιουργία, Διαγραφή, Μεταφόρτωση και Επεξεργασία Αρχείων

Εικόνα 12: Οθόνη ενός έργου για χρήστη με δικαιώματα ιδιοκτήτη σε αυτό

Ο λόγος που έχει νόημα ένα έργο είναι η διατήρηση αρχείων σε αυτό. Δίνουμε λοιπόν στον χρήστη ένα σύνολο λειτουργιών για την σωστή και εύκολη διατήρηση και επεξεργασία αυτών των αρχείων. Οι περισσότερες λειτουργίες των αρχείων βρίσκονται στην οθόνη του έργου στο οποίο ανήκουν. Όπως φαίνεται στην Εικόνα 12, η οθόνη του έργου περιέχει μία λίστα με τα αρχεία του. Η δημιουργία και η μεταφόρτωση ενός αρχείου ξεκινά καθώς ο χρήστης επιλέγει το σύμβολο συν (+) κάτω από την λίστα των αρχείων. Σε ένα νέο στοιχείο που εμφανίζεται με την μορφή αναδυόμενου παραθύρου, ο χρήστης μπορεί να εισάγει το όνομα του νέου αρχείου ή να μεταφορτώσει το αρχείο που επιθυμεί από το σύστημά του. Η διαγραφή αρχείων γίνεται αρχικά με την επιλογή τους από την λίστα. Έπειτα, επιλέγοντας την ‘Αφαίρεση Επιλεγμένων’, τα αρχεία απομακρύνονται από το σύστημα.



Εικόνα 13: Οθόνη επεξεργασίας αρχείου

Η πιο βασική λειτουργία των αρχείων είναι η επεξεργασία. Η διαδικασία της λειτουργίας αυτής περιγράφεται από την περίπτωση χρήσης που ακολουθεί. Σε αυτή την λειτουργία παρατηρούμε την χρήση του κειμενογράφου ACE που παρέχει ένα εύχρηστο περιβάλλον για την συγγραφή VHDL κώδικα όπως φαίνεται στην Εικόνα 13. Επίσης φαίνεται και η επιλογή θέματος που προσφέρουμε για τον κειμενογράφο που υποστηρίζει πάνω από 20 διαφορετικά θέματα εμφάνισης.

Περίπτωση Χρήσης: **Επεξεργασία Αρχείου**
Πρωταγωνιστής: Χρήστης
Προϋπόθεση: Ο χρήστης είναι συνδεμένος και βρίσκεται στην σελίδα ενός έργου.
Εκπλήρωση: Αποθηκεύονται οι αλλαγές που έκανε ο χρήστης στο αρχείο.
Βασικό Σενάριο:

1. Ο χρήστης πατάει με τον δείκτη ποντικιού στο όνομα του αρχείου που θέλει να επεξεργαστεί από την λίστα αρχείων.
2. Ο χρήστης μεταφέρεται στην σελίδα επεξεργασίας αρχείου.
3. Ο χρήστης πληκτρολογεί της αλλαγές που επιθυμεί στον κειμενογράφο.
4. Ο χρήστης επιλέγει ‘Αποθήκευση’.
5. Οι αλλαγές αποθηκεύονται και επιστρέφεται αντίστοιχο μήνυμα χωρίς την αλλαγή σελίδας ώστε να συνεχίσει η επεξεργασία.

Εναλλακτικά σενάρια:

1α. Ο χρήστης δεν έχει δικαιώματα επεξεργασίας σε αυτό το έργο.

1α1. Ο χρήστης μεταφέρεται στην σελίδα προβολής του αρχείου χωρίς δυνατότητες επεξεργασίας.

2α. Ο χρήστης προσπαθεί να μεταβεί από τον σύνδεσμο URL στην σελίδα επεξεργασίας αρχείου ενός έργου στο οποίο δεν έχει δικαιώματα προβολής.

2α1. Ο χρήστης μεταβαίνει σε μία κενή σελίδα επεξεργασίας όπου επιστρέφεται μήνυμα που ενημερώνει ότι είτε το αρχείο δεν υπάρχει ή ο χρήστης δεν έχει δικαιώματα επεξεργασίας σε αυτό.

4.1.3 Μεταγλώττιση και Προσομοίωση

Η μεταγλώττιση και η προσομοίωση κάθε έργου αποτελούν στοιχειώδεις λειτουργίες της εφαρμογής μας. Αυτές οι λειτουργίες αποτελούν εν τέλει την υλοποίηση των αρχικών στόχων της εφαρμογής. Οι λειτουργίες αυτές βρίσκονται στην οθόνη κάθε έργου με την δυνατότητα εκτέλεσής τους. Εστιάζοντας αρχικά στην μεταγλώττιση, βλέπουμε ότι η λειτουργία εφαρμόζεται ξεχωριστά σε κάθε αρχείο. Παρακάτω ακολουθεί η περίπτωση χρήσης για την μεταγλώττιση κάθε αρχείου.

Περίπτωση Χρήσης: **Μεταγλώττιση Αρχείου**
Πρωταγωνιστής: Χρήστης
Προϋπόθεση: Ο χρήστης είναι συνδεμένος και βρίσκεται στην σελίδα ενός έργου.
Εκπλήρωση: Η μεταγλώττιση του αρχείου ολοκληρώνεται και ο χρήστης ενημερώνεται για το γεγονός αυτό.
Βασικό Σενάριο:

1. Ο χρήστης επιλέγει το αρχείο που θέλει να μεταγλωττίσει από την λίστα αρχείων.
2. Ο χρήστης πατάει με τον δείκτη ποντικιού στην επιλογή ‘Μεταγλώττιση Επιλεγμένων’.
3. Το αρχείο τοποθετείτε στην ουρά αναμονής εργασιών και επιστρέφεται αντίστοιχο μήνυμα στον χρήστη.
4. Στην σελίδα του έργου εμφανίζεται μήνυμα δίπλα στο όνομα του αρχείου που ενημερώνει τον χρήστη ότι το αρχείο αναμένει μεταγλώττιση.
5. Η ουρά αναμονής φτάνει στην σειρά του αρχείου που δόθηκε και ολοκληρώνεται η μεταγλώττιση.
6. Ο χρήστης μεταβαίνοντας στην σελίδα του έργου βλέπει το μήνυμα δίπλα στο όνομα του αρχείου που τον ενημερώνει για την ολοκλήρωση της διαδικασίας.

Εναλλακτικά σενάρια:

1α. Ο χρήστης δεν έχει δικαιώματα επεξεργασίας σε αυτό το έργο.

1α1. Στην λίστα αρχείων δεν υπάρχει η δυνατότητα επιλογής και μεταγλώττισης των αρχείων.

6α. Η μεταγλώττιση του αρχείου αποτυγχάνει λόγο προβληματικού κώδικα.

6α1. Δίπλα στο όνομα του αρχείου εμφανίζεται σχετικό λάθος και επιλογή επέκτασης όπου συμπεριλαμβάνεται η έξοδος του μεταγλωττιστή με τις σχετικές αναφορές στο πρόβλημα που αντιμετώπισε.

6β. Το αρχείο επεξεργάζεται και ο κώδικάς του αλλάζει.

6β1. Δίπλα στο όνομα του αρχείου εμφανίζεται σχετικά μήνυμα ώστε να ενημερωθεί ο χρήστης ότι η τελευταία μεταγλώττιση δεν αντιστοιχεί στον νέο κώδικα.

Βλέπουμε λοιπόν ότι η διαδικασία που ακολουθεί ο χρήστης είναι τόσο απλή όσο η επιλογή ενός αρχείου και το πάτημα ενός κουμπιού. Από την άλλη όμως η εφαρμογή μεταβαίνει από διάφορα στάδια κρατώντας τον χρήστη σε συνεχή ενημέρωση. Έχοντας υπόψη ότι η μεταγλώττιση γίνεται από το εξωτερικό πρόγραμμα GHDL, γίνεται κατανοητό ότι η χρήση της ουράς αναμονής μέσω των σεναρίων εργασίας είναι απαραίτητη. Έτσι είναι απαραίτητες και οι καθυστερήσεις που προκύπτουν μεταξύ κάθε βήματος που πραγματοποιείται μέχρι την ολοκλήρωση της διαδικασίας. Το γεγονός αυτό ισχύει και για την λειτουργία προσομοίωσης του έργου. Η λειτουργία αυτή είναι προσβάσιμη αφού υπάρξει τουλάχιστον ένα μεταγλωττισμένο αρχείο. Η περίπτωση χρήσης που αντιστοιχεί στην προσομοίωση ενός έργου περιγράφεται παρακάτω.

Περίπτωση Χρήσης: **Προσομοίωση Έργου**
Πρωταγωνιστής: Χρήστης
Προϋπόθεση: Ο χρήστης είναι συνδεμένος και βρίσκεται στην σελίδα ενός έργου.
Εκπλήρωση: Η προσομοίωση έργου ολοκληρώνεται.
Βασικό Σενάριο:

1. Ο χρήστης επιλέγει την αρχιτεκτονική που θα προσομοιωθεί από τα διαθέσιμα μεταγλωττισμένα αρχεία του έργου.
2. Ο χρήστης επιλέγει τις πρόσθετες επιλογές προσομοίωσης που επιθυμεί.
3. Ο χρήστης πατάει με τον δείκτη ποντικιού στην επιλογή ‘Προσομοίωση Έργου’.
4. Το έργο τοποθετείται στην ουρά αναμονής εργασιών και επιστρέφεται σχετικό μήνυμα.
5. Η ουρά αναμονής φτάνει στην σειρά του αρχείου που δόθηκε και ολοκληρώνεται η προσομοίωση.
6. Ο χρήστης μεταβαίνοντας στην σελίδα του έργου βλέπει την επιλογή προβολής κυματομορφών των αποτελεσμάτων προσομοίωσης.

Εναλλακτικά σενάρια:

1α. Ο χρήστης δεν έχει δικαιώματα επεξεργασίας σε αυτό το έργο.

1α1. Οι επιλογές προσομοίωσης έργου δεν εμφανίζονται στον χρήστη.

1β. Το έργο δεν περιέχει μεταγλωττισμένα αρχεία.

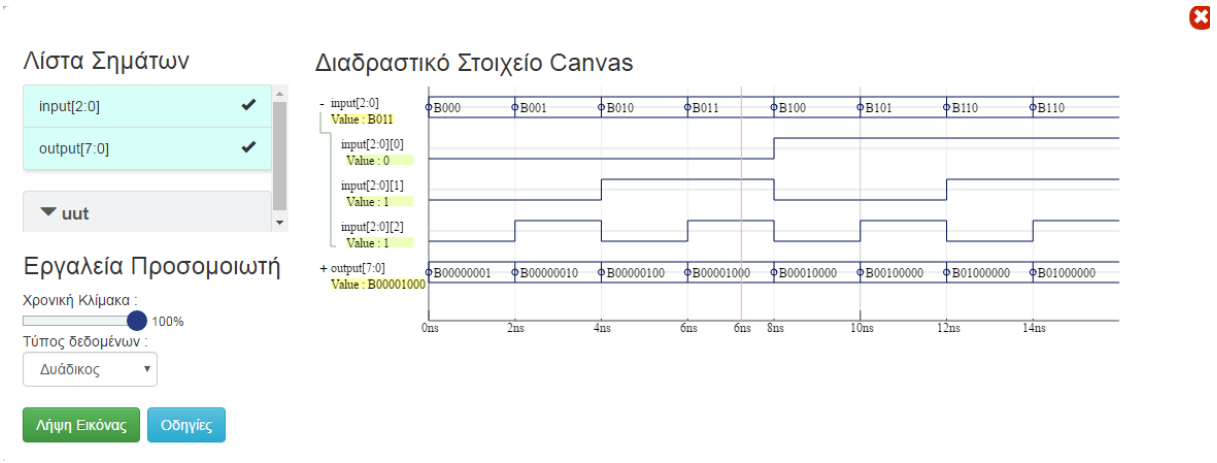
1β1. Οι επιλογές προσομοίωσης έργου δεν εμφανίζονται στον χρήστη.

4.1.4 Λειτουργίες του Εργαλείου Σχεδίασης Κυματομορφών

Αφού ολοκληρωθεί η προσομοίωση ενός έργου, γίνεται δυνατή η προβολή των αποτελεσμάτων του ως κυματομορφές. Έχοντας αναλύσει τον τρόπο υλοποίησης των λειτουργιών του εργαλείου σχεδίασης κυματομορφών, μένει πλέον να αναφέρουμε τον τρόπο που ο χρήστης αλληλεπιδρά με τις λειτουργίες που προσφέρονται.

Όπως φαίνεται στην Εικόνα 14, οι λειτουργίες του εργαλείου παρουσιάζονται στον χρήστη ως ένα σύνολο επιλογών και ένα διαδραστικό στοιχείο ‘canvas’. Εξωτερικά του διαδραστικού στοιχείου ο χρήστης έχει πρόσβαση στην επιλογή των σημάτων που θα εμφανιστούν, την αλλαγή της χρονικής κλίμακας, την μεταβίβαση σε διαφορετικό τύπο που

εμφανίζονται τα δεδομένα, την λήψη στιγμιότυπου ως εικόνα και την εμφάνιση ή απόκρυψη οδηγιών χρήσης του εργαλείου. Εσωτερικά του διαδραστικού στοιχείου, ο χρήστης μπορεί να χρησιμοποιήσει τους δείκτες επιπλέων πληροφοριών, να επεκτείνει ή συμπυκνώσει σήματα με μεγάλο μέγεθος καθώς και να μετακινηθεί στον χρονικό άξονα όταν η χρονική κλίμακα δεν περιλαμβάνει το σύνολο του χρόνου προσομοίωσης. Παρακάτω θα δούμε μία περίπτωση χρήσης που αξιοποιεί πολλές από τις λειτουργίες του εργαλείου.



Εικόνα 14: Οθόνη του εργαλείου σχεδίασης κυματομορφών

Περίπτωση Χρήσης: **Λήψη Εικόνας Κυματομορφής με Τιμή του Σήματος σε μία Χρονική Στιγμή**
Πρωταγωνιστής: Χρήστης
Προϋπόθεση: Ο χρήστης είναι συνδεμένος και βρίσκεται στην σελίδα ενός έργου.
Εκπλήρωση: Ο χρήστης μεταφορτώνει την εικόνα κυματομορφών όπου φαίνονται οι τιμές κάθε σήματος που επέλεξε σε έναν συγκεκριμένο χρόνο προσομοίωσης.
Βασικό Σενάριο:

1. Ο χρήστης επιλέγει το αρχείο VCD που θα εξετάσει
2. Ο χρήστης πατάει με τον δείκτη ποντικιού στην επιλογή ‘Προβολή Κυματομορφών’.
3. Εμφανίζεται το στοιχείο που περιέχει το εργαλείο σχεδίασης κυματομορφών και φορτώνει τα δεδομένα του αρχείου VCD.
4. Ο χρήστης επιλέγει τα σήματα που τον ενδιαφέρουν από την λίστα σημάτων.
5. Ο χρήστης επιλέγει τον τύπο δεδομένων που θα χρησιμοποιήσει το εργαλείο.
6. Ο χρήστης πατάει με τον δείκτη ποντικιού στον χρόνο προσομοίωσης που τον ενδιαφέρει.
7. Δημιουργείται μόνιμος δείκτης επιπλέον πληροφοριών στο σημείο που πάτησε ο χρήστης.
8. Ο χρήστης πατάει την επιλογή ‘Λήψη Εικόνας’.
9. Ένα στιγμιότυπο του στοιχείου ‘canvas’ μεταφορτώνεται στο σύστημα του χρήστη ως εικόνα.

Εναλλακτικά σενάρια:

1α. Ο χρήστης δεν έχει δικαιώματα επεξεργασίας σε αυτό το έργο.

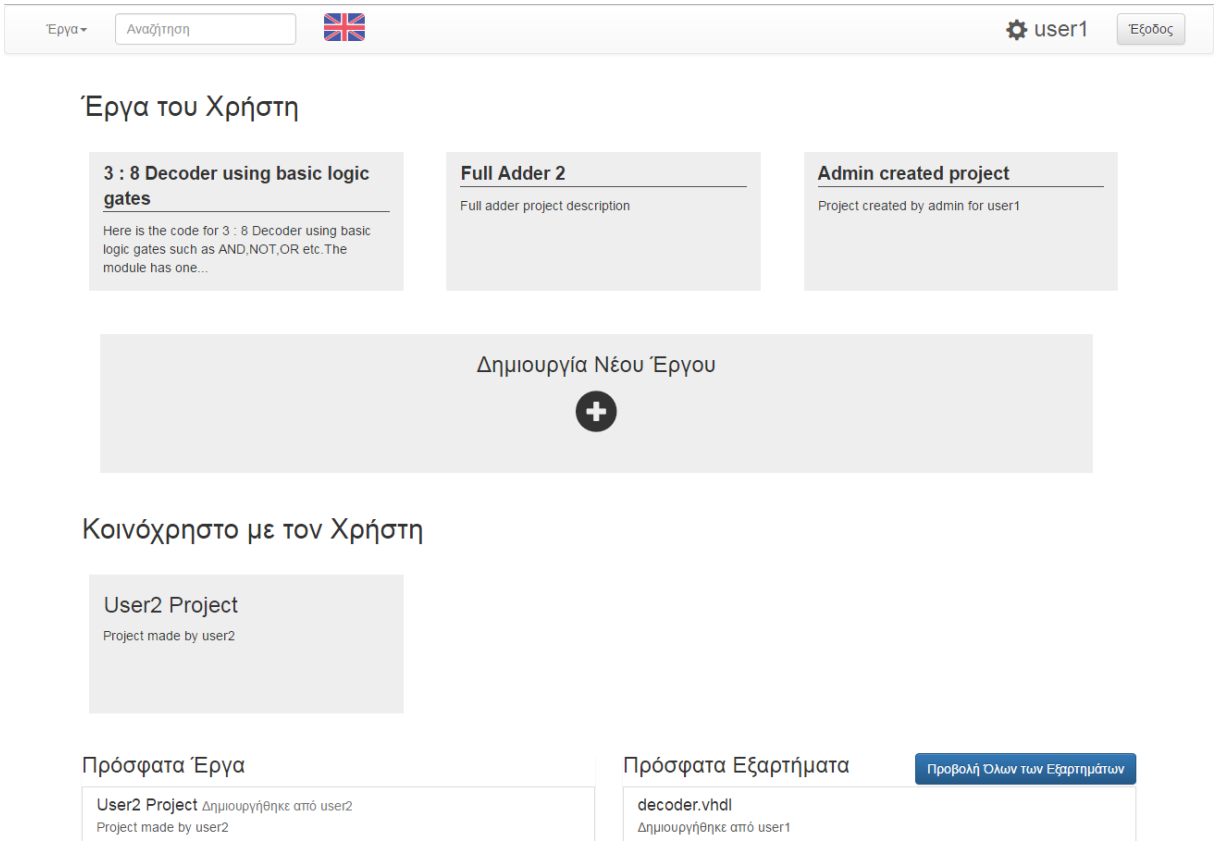
1α1. Οι επιλογές προβολής αποτελεσμάτων προσομοίωσης δεν εμφανίζονται στον χρήστη.

6α. Οι διαστάσεις κάθε παλμού είναι πολύ μικρές και ο χρήστης δεν μπορεί να βρει τον χρόνο που τον ενδιαφέρει.

6α1. Ο χρήστης μειώνει την χρονική κλίμακα ώστε το μέγεθος κάθε παλμού να είναι αρκετά μεγάλο.

6α2. Ο χρήστης μετακινεί τις κυματομορφές στον χρονικό άξονα ώστε να εμφανίζεται η χρονική στιγμή που τον ενδιαφέρει.

4.1.5 Δυνατότητες Διαμοίρασης Έργων



Εικόνα 15: Αρχική οθόνη συνδεμένου χρήστη

Όπως αναφέραμε στο πρώτο κεφάλαιο, το διαδικτυακό περιβάλλον της εφαρμογής μας επιτρέπει την εισαγωγή συλλογικότητας στην ανάπτυξη των έργων. Σε κάθε έργο η συλλογικότητα μεταφράζεται σε δύο λειτουργίες. Η πρώτη λειτουργία είναι αυτή της απλής προσπέλασης του έργου και των αρχείων του από άλλους χρήστες. Η λειτουργία αυτή βρίσκεται εν μέρει στην δημιουργία και επεξεργασία του έργου όπου δίνεται η δυνατότητα στον χρήστη να θέσει το έργο του ως δημόσιο ή ιδιωτικό. Ένα ιδιωτικό έργο δεν είναι προσβάσιμο από άλλους χρήστες και εάν ένας άλλος χρήστης προσπαθήσει να επισκεφτεί την διεύθυνση που αντιστοιχεί σε ένα τέτοιο έργο θα λάβει αντίστοιχο μήνυμα ενημέρωσης. Αντίθετα ένα δημόσιο έργο είναι προσβάσιμο από κάθε χρήστη και επισκέπτη. Ένα τέτοιο έργο εμφανίζεται στα πρόσφατα έργα και μπορεί να βρεθεί από την μπάρα αναζήτησης. Όταν ένας επισκέπτης επισκεφτεί την διεύθυνση του έργου, μπορεί να δει το όνομα, την περιγραφή και τα αρχεία του έργου χωρίς όμως καμία δυνατότητα επεξεργασίας ή εκτέλεσης λειτουργιών σε αυτό. Οι δυνατότητες αυτές βρίσκονται στην δεύτερη λειτουργία διαμοίρασης έργων. Η

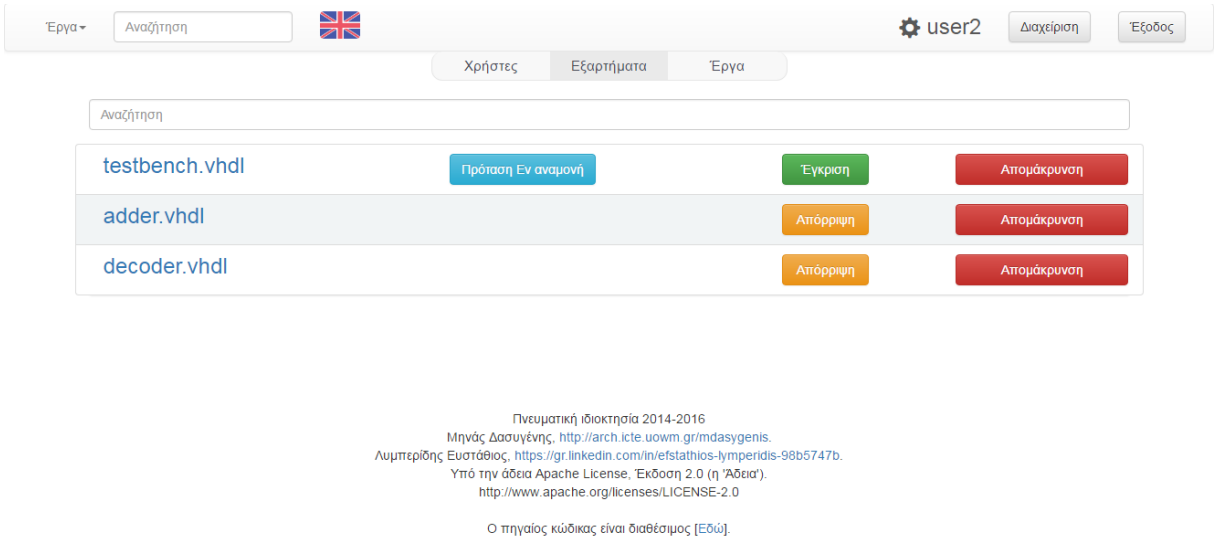
λειτουργία αυτή επιτρέπει την εισαγωγή λογαριασμών χρηστών ως συντάκτες του έργου. Έτσι προσφέρει σε άλλους χρήστες την δυνατότητα να συμβάλουν στην ανάπτυξή του έχοντας πρόσβαση σε όλες τις λειτουργίες του έργου εκτός της επεξεργασίας και κατάργησής του. Σε κάθε έργο υπάρχει μία λίστα που επιδεικνύει το όνομα του ιδιοκτήτη και των συντακτών του. Επίσης, εάν ένας χρήστης είναι συντάκτης σε ένα ή περισσότερα έργα, έχει διαθέσιμη μία ξεχωριστή λίστα στην αρχική του σελίδα με τα κοινόχρηστα έργα όπως φαίνεται στην Εικόνα 15. Οι λειτουργίες αυτές είναι συμπληρωματικές αυτών που αναφέραμε παραπάνω. Είδαμε ήδη πως επηρεάζουν κάποιες περιπτώσεις χρήσης επιτρέποντας διαφορετικά δικαιώματα πρόσβασης και την εισαγωγή επιλογών στην δημιουργία και επεξεργασία των έργων. Στο επόμενο υποκεφάλαιο θα δούμε κάποιες λειτουργίες που επεκτείνουν ακόμη περισσότερο την συλλογικότητα που προσφέρει η εφαρμογή μας με μία διαφορετική προσέγγιση.

4.2 Λειτουργίες Εξαρτημάτων

Αναπτύσσοντας μία περιγραφή VHDL, είναι πολλές φορές βολική η χρήση εξαρτημάτων. Τα εξαρτήματα χρησιμοποιούνται συνήθως με παρόμοιο τρόπο που χρησιμοποιείται μία βιβλιοθήκη ή συνάρτηση και αποτελούν ξεχωριστά κομμάτια κώδικα που εκτελούν μία συγκεκριμένη λειτουργία. Πολλές φορές κάποια εξαρτήματα αποτελούν λειτουργίες που εμφανίζονται σε έναν μεγάλο αριθμό προγραμμάτων. Η δυνατότητα απόκτησης τέτοιων εξαρτημάτων και η άμεση προσθήκη τους σε ένα έργο επιτρέπει στον προγραμματιστή να επικεντρωθεί στο κομμάτι που κάνει το πρόγραμμά του διαφορετικό. Έτσι αναπτύξαμε στην εφαρμογή μας τις λειτουργίες που παρουσιάζονται παρακάτω για να δώσουμε την ικανότητα στον κάθε χρήστη να μοιραστεί αλλά και να αποκτήσει από άλλους χρήστες χρήσιμα εξαρτήματα. Παρακάτω θα δούμε την διαδικασία ανάρτησης και λήψης εξαρτημάτων από έναν χρήστη και ενημέρωσης εξαρτημάτων. Θα εξηγήσουμε επίσης τις λειτουργίες που επιτρέπουν την προβολή εξαρτημάτων με μία βαθμονομημένη διάταξη.

4.2.1 Ανάρτηση και Λήψη Εξαρτημάτων

Το κάθε εξάρτημα δημοσιοποιείται από κάποιον χρήστη. Καθώς ο χρήστης αναπτύσσει ένα εξάρτημα, μπορεί να δημιουργήσει ένα ξεχωριστό αρχείο γι’ αυτό μέσα στο έργο του. Έτσι το εξάρτημά του γίνεται μία αυτόνομη οντότητα. Έπειτα ο χρήστης μπορεί να δημοσιεύσει αυτό το αρχείο ως εξάρτημα από την σελίδα του.



Εικόνα 16: Οθόνη διαχείρισης εξαρτημάτων

Καθώς η αυθαίρετη δημοσίευση αρχείων από τους χρήστες θα μπορούσε να οδηγήσει σε αθέμητη χρήση της εφαρμογής μας, το εξάρτημα δεν γίνεται αμέσως ορατό προς τους υπόλοιπους χρήστες κατά την δημοσίευσή του. Γίνεται όμως διαθέσιμο στην λίστα εξαρτημάτων που προσφέρεται στους διαχειριστές. Εκεί τα εξαρτήματα που μόλις δημοσιοποιήθηκαν θα τοποθετούνται στην αρχή της λίστας και θα έχουν την δυνατότητα έγκρισης όπως φαίνεται στην Εικόνα 16. Η έγκριση ενός εξαρτήματος το καθιστά προσβάσιμο από κάθε χρήστη καθώς και του επιτρέπει να εμφανίζεται τόσο στην λίστα όλων των εξαρτημάτων όσο και σαν αποτέλεσμα αναζήτησης. Παρακάτω αναφέρουμε την περίπτωση χρήσης για την έγκριση ενός εξαρτήματος.

Περίπτωση Χρήσης: **Έγκριση Εξαρτήματος**
Πρωταγωνιστής: Διαχειριστής
Προϋπόθεση: Ο διαχειριστής είναι συνδεμένος και βρίσκεται στην σελίδα διαχείρισης εξαρτημάτων.
Εκπλήρωση: Το εξάρτημα εγκρίνεται και γίνεται διαθέσιμο στους υπόλοιπους χρήστες.
Βασικό Σενάριο:

1. Ο διαχειριστής πατάει με τον δείκτη του ποντικιού στο όνομα του εξαρτήματος που τον ενδιαφέρει.
2. Ο διαχειριστής μεταφέρεται στην σελίδα επεξεργασίας του εξαρτήματος.
3. Ο διαχειριστής πατάει με τον δείκτη του ποντικιού στην επιλογή ‘Έγκριση’.

Εναλλακτικά σενάρια:

2α. Ο διαχειριστής παρατηρεί προβληματικό κώδικα στο εξάρτημα.

2α1. Ο διαχειριστής διορθώνει τον κώδικα.

2α2. Ο διαχειριστής επιλέγει αποθήκευση.

2β. Ο διαχειριστής παρατηρεί ότι το εξάρτημα δεν είναι έγκυρο και πρέπει να απορριφθεί.

2β1. Ο διαχειριστής επιλέγει ‘Διαχείριση’.

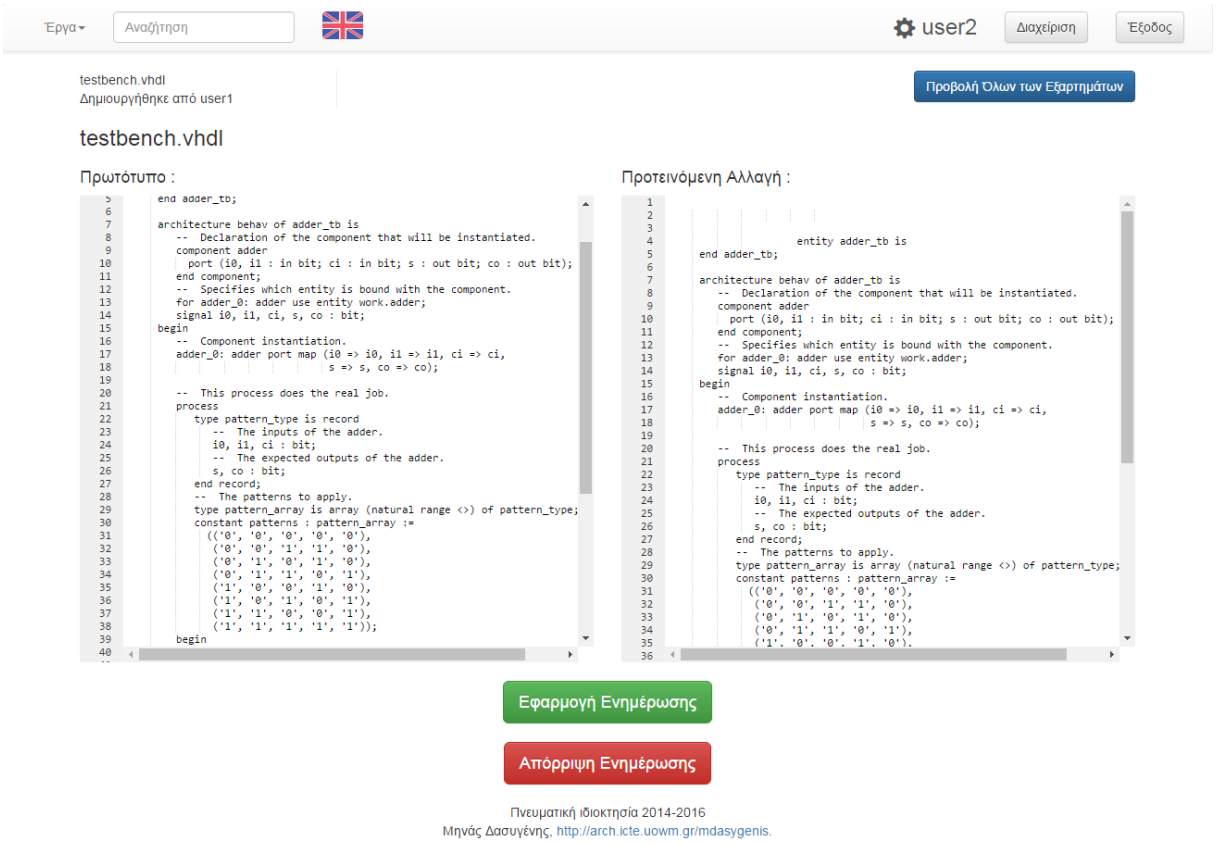
2β2. Ο διαχειριστής επιστρέφει στην οθόνη διαχείρισης και επιλέγει ‘Εξαρτήματα’.

2β3. Ο διαχειριστής επιλέγει ‘Απομάκρυνση’ στο εξάρτημα που τον ενδιαφέρει.

Με παρόμοιο τρόπο γίνεται η απόρριψη ενός εξαρτήματος που έχει εγκριθεί. Η απόρριψη θα αποκρύψει το εξάρτημα αυτό από τους χρήστες και θα το αναγκάσει να αναμένει νέα έγκριση. Όταν ένα εξάρτημα έχει εγκριθεί, γίνεται λοιπόν διαθέσιμο στους χρήστες. Ο κάθε χρήστης μπορεί να προσπελάσει τον κώδικά του και μέσα από την οθόνη προβολής του εξαρτήματος να επιλέξει ένα από τα έργα του ώστε να το προσθέσει σε αυτό. Τότε ένα αντίγραφο του εξαρτήματος θα δημιουργηθεί μέσα στο έργο του χρήστη επιτρέποντας την χρήση αλλά και τροποποίηση του εξαρτήματος.

4.2.2 Διαδικασία Ενημερώσεων Εξαρτημάτων

Στο προηγούμενο υποκεφάλαιο είδαμε τον τρόπο που οι χρήστες μπορούν να μοιραστούν εξαρτήματα. Ένα βασικό στοιχείο όμως των εξαρτημάτων είναι η συντήρηση και βελτιστοποίησή τους. Ο τελικός σκοπός είναι να ανταλλαγή δοκιμασμένου και βελτιστοποιημένου κώδικα για συγκεκριμένες λειτουργίες.



Εικόνα 17: Οθόνη επεξεργασίας προτεινόμενης ενημέρωσης

Αυτό προϋποθέτει την λειτουργία ενημέρωσης. Καθώς ο συγγραφέας ενός εξαρτήματος αναπτύσσει το έργο του, μπορεί να διαπιστώσει κάποιες απαραίτητες επεκτάσεις ή διορθώσεις στο εξάρτημα. Σε αυτή την περίπτωση αφού κάνει τις τροποποιήσεις μπορεί να επιλέξει ξανά την δημοσίευση του εξαρτήματος οπότε θα γίνει ενημέρωση του υπάρχοντος. Τότε, στην σελίδα διαχείρισης εξαρτημάτων, το εξάρτημα αυτό θα τοποθετηθεί σε προτεραιότητα μετά από αυτά που περιμένουν έγκριση και θα έχει την ένδειξη ‘Πρόταση Εν αναμονή’. Έτσι ο διαχειριστής θα μπορέσει να μεταβεί στην οθόνη επεξεργασίας προτεινόμενης ενημέρωσης και

να επιτρέψει ή απορρίψει την ενημέρωση του χρήστη όπως φαίνεται στην Εικόνα 17. Εάν η ενημέρωση εφαρμοστεί ή εάν ένας διαχειριστής επεξεργαστεί ένα εξάρτημα και εφαρμόσει δική του αλλαγή σε αυτό, οι χρήστες που έχουν εισάγει το εξάρτημα σε ένα έργο του θα ενημερωθούν. Στην λίστα αρχείων του έργου που υπάρχει το εξάρτημα θα τους δοθεί η ένδειξη ότι υπάρχει διαθέσιμη ενημέρωση γι' αυτό και η επιλογή να μεταβούν απευθείας στην προβολή του εξαρτήματος για την εισαγωγή του ενημερωμένου αρχείου στο έργο τους. Παρακάτω θα δούμε λεπτομερώς την περίπτωση χρήσης για την εφαρμογή μίας ενημέρωσης εξαρτήματος.

Περίπτωση Χρήσης: **Εφαρμογή Ενημέρωσης Εξαρτήματος**
Πρωταγωνιστής: Διαχειριστής
Προϋπόθεση: Ο διαχειριστής είναι συνδεμένος και βρίσκεται στην σελίδα διαχείρισης εξαρτημάτων.
Εκπλήρωση: Οι αλλαγές της ενημέρωσης περνάνε στο εξάρτημα.
Βασικό Σενάριο:

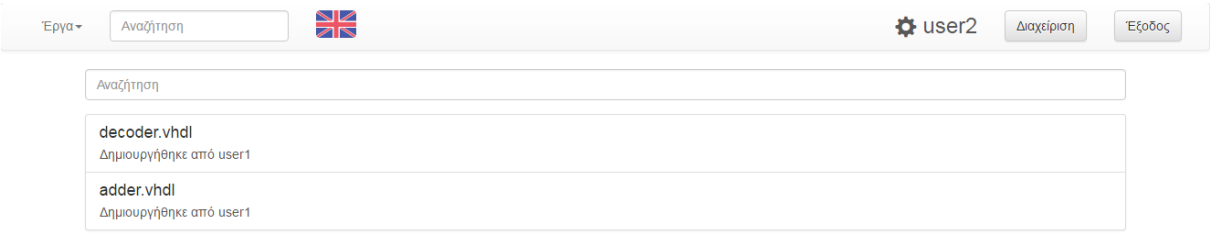
1. Ο διαχειριστής πατάει με τον δείκτη του ποντικιού στην επιλογή 'Πρόταση Εν αναμονή' στο εξάρτημα που τον ενδιαφέρει.
2. Ο διαχειριστής μεταφέρεται στην σελίδα επεξεργασίας προτεινόμενης ενημέρωσης του εξαρτήματος.
3. Ο διαχειριστής πατάει με τον δείκτη του ποντικιού στην επιλογή 'Εφαρμογή Ενημέρωσης'.

Εναλλακτικά σενάρια:

2α. Ο διαχειριστής παρατηρεί προβληματικό κώδικα στην προτεινόμενη ενημέρωση.
2α1. Ο διαχειριστής διορθώνει τον κώδικα ενημέρωσης.
2β. Ο διαχειριστής παρατηρεί ότι η προτεινόμενη ενημέρωση είναι προβληματική.
2β1. Ο διαχειριστής επιλέγει 'Απόρριψη Ενημέρωσης'.
2β2. Η ενημέρωση απορρίπτεται και δεν εμφανίζεται ξανά.

4.2.3 Προβολή και Βαθμονόμηση Εξαρτημάτων

Τα εξαρτήματα που έχουν εγκριθεί είναι διαθέσιμα σε κάθε χρήστη. Η πρώτη επαφή του χρήστη με αυτά είναι στην αρχική του σελίδα όπου του προσφέρεται μία λίστα με τα πέντε πιο πρόσφατα εξαρτήματα. Πάνω από την λίστα βρίσκεται η επιλογή για την προβολή όλων των εξαρτημάτων.



Εικόνα 18: Οθόνη προβολής όλων των εξαρτημάτων που έχουν εγκριθεί

Η επιλογή αυτή μεταφέρει τον χρήστη στην σελίδα όπου προσφέρεται μία λίστα των εξαρτημάτων που έχουν εγκριθεί, όπως βλέπουμε στην Εικόνα 18. Καθώς με την ευρεία χρήση της εφαρμογής η λίστα αυτή προβλέπεται να γίνει αρκετά μεγάλη, δημιουργείται η ανάγκη για

την ταξινόμησή της. Η ταξινόμηση αυτή θα μπορούσε να είναι τυχαία, με βάση τον χρόνο δημιουργίας ή αλφαβητική. Αυτές οι επιλογές όμως δεν προσφέρουν κάποια επιπλέον πληροφορία, σημαντική για τον χρήστη. Έτσι αποφασίσαμε να κάνουμε την ταξινόμηση με βάση τον αριθμό προσθήκης στα έργα άλλων χρηστών. Όταν ένας χρήστης προσθέτει ένα εξάρτημα στο έργο του, ο βαθμός του εξαρτήματος αυξάνεται κατά ένα. Στην λίστα προβολής όλων των εξαρτημάτων έχουμε λοιπόν την εφαρμογή μίας φθίνουσας ταξινόμησης με βάση τον βαθμό που συγκέντρωσε το καθένα. Ως αποτέλεσμα τα χρησιμότερα εξαρτήματα εμφανίζονται πρώτα κι έτσι η πιθανότητα να βρει άμεσα το εξάρτημα που θέλει ο χρήστης αυξάνεται. Εκτός αυτής της λειτουργίας προσφέρεται λόγω του αναμενόμενου μεγέθους της λίστας η σελιδοποίηση ανά είκοσι (20) εγγραφές. Επίσης δίνεται στον χρήστη η δυνατότητα εφαρμογής ενός φίλτρου αναζήτησης που αποκρύπτει δυναμικά τα εξαρτήματα που δεν ταιριάζουν με αυτό.

Έτσι καταλήγουμε στην ολοκλήρωση των λειτουργιών που αποτελούν το σύστημα των εξαρτημάτων. Το σύστημα αυτό όπως είδαμε προσφέρει στην εφαρμογή μας μία επιπλέον διάσταση, επιτρέποντας στους χρήστες την ασύγχρονη διαμοίραση επαναχρησιμοποιήσιμου κώδικα. Μέσα από αυτό το σύστημα όμως είδαμε και μερικές διαφορές μεταξύ των διάφορων χρηστών. Στο επόμενο υποκεφάλαιο θα αναπτύξουμε περαιτέρω αυτές τις διαφορές εξηγώντας τα δικαιώματα πρόσβασης στις διάφορες λειτουργίες για κάθε επίπεδο χρηστών.

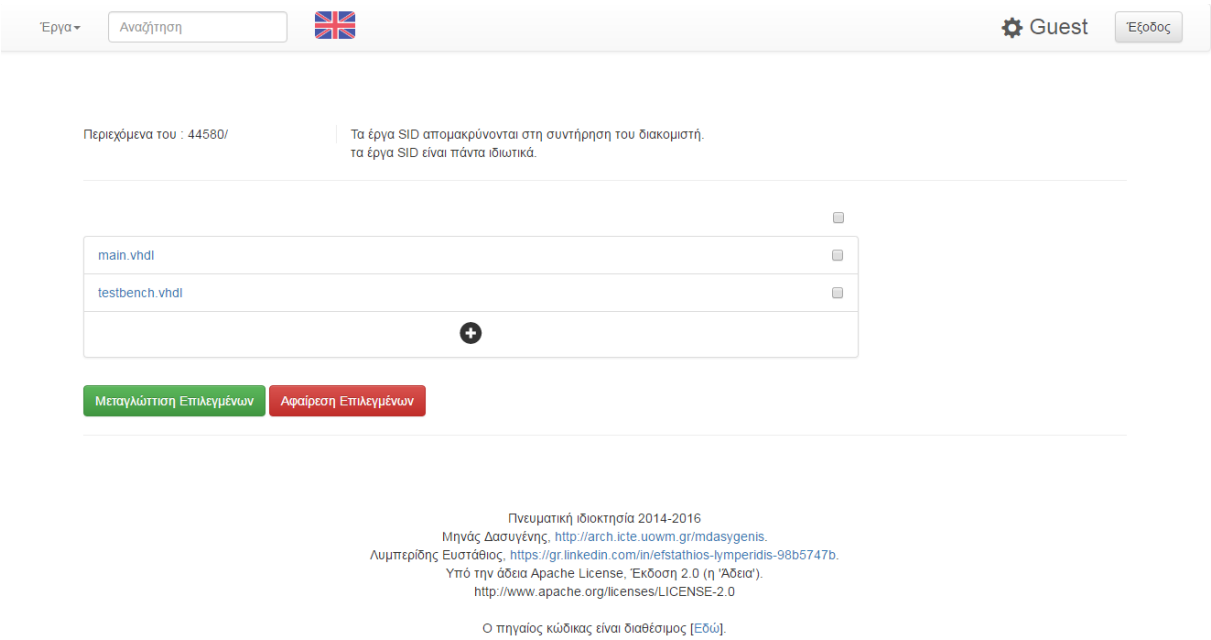
4.3 Επίπεδα Χρηστών

Στην εφαρμογή που αναπτύξαμε υπάρχει εμφωλευμένο ένα πλήρες σύστημα διαχείρισης χρηστών. Ένα τέτοιο σύστημα είναι απαραίτητο για την διατήρηση και συντήρηση μίας εφαρμογής που χρησιμοποιεί λογαριασμούς χρηστών. Πόσο μάλλον όταν οι λειτουργίες που προσφέρει είναι ριζικά δεμένες με την χρήση ξεχωριστού λογαριασμού από κάθε χρήστη. Έτσι, θα εξηγήσουμε σε αυτό το υποκεφάλαιο τα τρία διαφορετικά επίπεδα χρηστών καθώς και τις δυνατότητες που επιτρέπει το καθένα. Αξίζει να αναφέρουμε σε αυτό το σημείο ότι υπάρχουν κάποιες λειτουργίες που είναι διαθέσιμες σε όλους τους επισκέπτες. Αυτές οι λειτουργίες είναι η προβολή δημόσιων έργων και των αρχείων τους με την απευθείας πλοήγηση στον σύνδεσμό τους. Η λειτουργίες αυτές δεν απαιτούν κάποια πρόσβαση έτσι ώστε να είναι άμεσα διαθέσιμη η διαμοίραση έργου και κώδικα σε οποιονδήποτε επισκέπτη. Παρακάτω, θα δούμε αρχικά τους περιορισμούς των επισκεπτών, έπειτα τις λειτουργίες χρηστών και τέλος τα συστήματα που προσφέρονται στους διαχειριστές για την επίτευξη του σκοπού τους.

4.3.1 Δυνατότητες Επισκεπτών

Το επίπεδο επισκεπτών βρίσκεται στην εφαρμογή μας με σκοπό την επίδειξη των βασικών λειτουργιών που προσφέρει. Είναι επίσης ένα επίπεδο που δίνει την δυνατότητα σε έναν επισκέπτη να αξιοποιήσει τις λειτουργίες της εφαρμογής για μία γρήγορη μεταγλώττιση και προσομοίωση του κώδικά του, χωρίς την ανάγκη δημιουργίας λογαριασμού και έργου. Έτσι αντί για την διατήρηση ενός μόνιμου λογαριασμού χρήστη, ένας επισκέπτης μπορεί να λάβει ένα μοναδικό αριθμό συνόδου (SessionID).

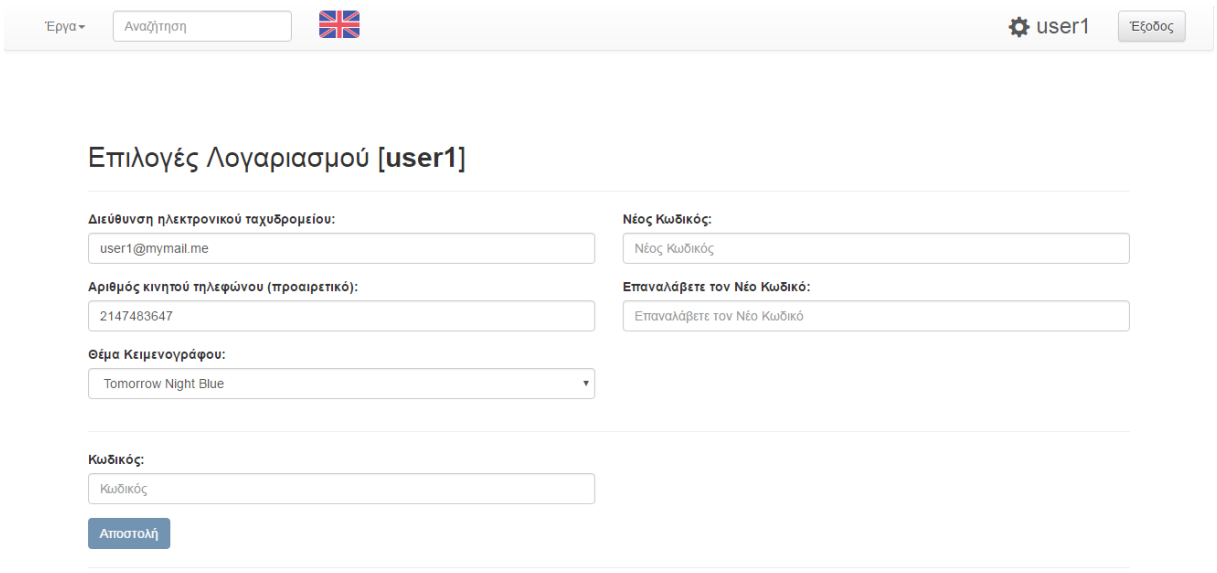
Κάνοντας είσοδο στην εφαρμογή με αυτόν τον αριθμό, βλέπουμε στην Εικόνα 19 ότι ο επισκέπτης έχει διαθέσιμο ένα έργο στο οποίο μπορεί να δημιουργήσει ή μεταφορτώσει αρχεία τα οποία μπορεί μετά να επεξεργαστεί. Του δίνεται επίσης η δυνατότητα να μεταγλωττίσει αυτά τα αρχεία και να προσομοιώσει τον κώδικά του. Τέλος έχει πρόσβαση στην ανάλυση των αποτελεσμάτων από το εργαλείο σχεδίασης κυματομορφών. Ο επισκέπτης δεν έχει πρόσβαση σε άλλες λειτουργίες του συστήματος. Επίσης τα αρχεία και ο αριθμός συνόδου του επισκέπτη χάνονται στην επανεκκίνηση του εξυπηρετητή καθώς βρίσκονται σε προσωρινό κατάλογο. Γίνεται έτσι φανερό ότι ο επισκέπτης έχει πρόσβαση σε περιορισμένες λειτουργίες για περιορισμένο χρονικό διάστημα που είναι όμως αρκετά για μία συγκεκριμένη εργασία ή για την εξερεύνηση της εφαρμογής πριν την δημιουργία λογαριασμού.



Εικόνα 19: Αρχική οθόνη επισκέπτη

4.3.2 Δυνατότητες Χρηστών

Ένας βασικός χρήστης της εφαρμογής ξεκινάει δημιουργώντας τον λογαριασμό του. Στην συνέχεια λαμβάνει ένα μήνυμα στην διεύθυνση ηλεκτρονικού ταχυδρομείου που παρέχει κατά την δημιουργία λογαριασμού με τον κωδικό επιβεβαίωσης. Αφού συνδεθεί ο χρήστης, του ζητείται ο κωδικός επιβεβαίωσης για την ενεργοποίηση του λογαριασμού. Στο τέλος της διαδικασίας ο χρήστης έχει ενεργοποιηθεί και όλες οι βασικές λειτουργίες είναι πλέον διαθέσιμες γι' αυτόν.

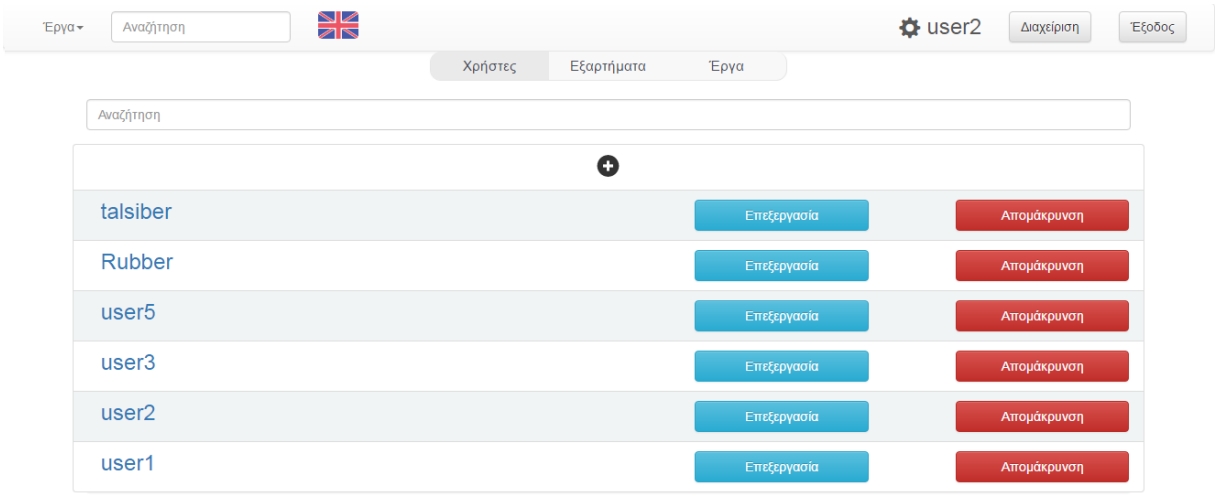


Εικόνα 20: Οθόνη επιλογών λογαριασμού χρήστη

Ένας συνδεμένος χρήστης έχει πλέον την δυνατότητα δημιουργίας έργων στα οποία του παρέχονται οι λειτουργίες που εξετάσαμε σε προηγούμενο υποκεφάλαιο. Επίσης ο συνδεμένος χρήστης μπορεί να γίνει συντάκτης σε έργα άλλων χρηστών. Τέλος ο συνδεμένος χρήστης μπορεί να δημοσιεύσει και να λάβει εξαρτήματα καθώς και να προβάλει αυτά που έχουν εγκριθεί. Ένας χρήστης μπορεί επίσης να αλλάξει πολλές από τις επιλογές που έδωσε κατά την δημιουργία του λογαριασμού του. Αυτό γίνεται πατώντας στο γρανάζι δίπλα από το όνομά του στην μπάρα πλοήγησης. Έτσι ο χρήστης θα μεταβεί στην σελίδα επιλογών λογαριασμού χρήστη όπως φαίνεται στην Εικόνα 20. Εκεί μπορεί να επεξεργαστεί τις λεπτομέρειες του λογαριασμού του και να αποθηκεύσει τις αλλαγές επιβεβαιώνοντας με τον κωδικό του.

4.3.3 Δυνατότητες Διαχειριστών

Οι διαχειριστές της εφαρμογής είναι υπεύθυνοι για την ομαλή λειτουργία της. Έτσι προσφέρονται σε αυτούς κάποιες επιπλέον δυνατότητες που τους επιτρέπουν την επίτευξη του σκοπού τους. Ο κάθε λογαριασμός διαχειριστή είναι και λογαριασμός χρήστη, έχοντας έτσι πρόσβαση σε όλες τις λειτουργίες συνδεμένων χρηστών. Επιπλέον όμως ο κάθε διαχειριστής θεωρείται και ιδιοκτήτης κάθε έργου, επιφέροντας έτσι πλήρες δικαιώματα σε αυτά. Ο κύριος διαχωρισμός τους όμως από τους συνδεμένους χρήστες βρίσκεται στην πρόσβαση των συστημάτων διαχείρισης.



Εικόνα 21: Οθόνη διαχείρισης χρηστών

Αρχικά, με την μετάβαση στην σελίδα διαχείρισης, έχουμε το σύστημα διαχείρισης χρηστών, όπως φαίνεται στην Εικόνα 21. Στο προηγούμενο υποκεφάλαιο είδαμε ήδη το σύστημα διαχείρισης εξαρτημάτων και με παρόμοιο τρόπο έχουμε το σύστημα διαχείρισης έργων. Τα συστήματα αυτά αποσκοπούν στην προσφορά λειτουργιών για την εύκολη και γρήγορη εύρεση ενός στοιχείου και την εφαρμογή της επιθυμητής ενέργειας σε αυτό. Έτσι, αποτελούνται από λίστες των αντίστοιχων στοιχείων, με φίλτρα αναζήτησης και επιλεκτική ταξινόμηση όταν υπάρχει νόημα σε αυτό. Για παράδειγμα, η λίστα εξαρτημάτων ταξινομείται σε τρία επίπεδα όπου δίνεται αρχικά προτεραιότητα εμφάνισης στα εξαρτήματα που αναμένουν έγκριση, μετά εμφανίζονται αυτά που αναμένουν εφαρμογή ενημέρωσης και τέλος εμφανίζονται τα υπόλοιπα. Επίσης στις λίστες αυτές, κάθε στοιχείο περιέχει τις πιθανές ενέργειες που μπορούν να γίνουν σε αυτό ενώ επιλέγοντας το ίδιο το στοιχείο πατώντας στο όνομά του, μεταφέρει τον διαχειριστή στην σελίδα προβολής του στοιχείου. Έτσι προσφέρονται στους λογαριασμούς διαχειριστών ένα σύνολο λειτουργιών που τους επιτρέπουν την συντήρηση και διατήρηση της ομαλής λειτουργίας της εφαρμογής μας.

4.4 Σύνοψη Κεφαλαίου

Σε αυτό το κεφάλαιο είδαμε τις λειτουργίες που προσφέρει η εφαρμογή μας. Εξηγήσαμε τον τρόπο που αλληλεπιδρά ένας χρήστης με την εφαρμογή για την επίτευξη διάφορων λειτουργών και αναφέραμε πολλές από τις περιπτώσεις χρήσης που μπορεί να αντιμετωπίσει. Αναφέραμε επίσης τα διαφορετικά επίπεδα χρηστών και τις δυνατότητες που προσφέρει η εφαρμογή στο καθένα από αυτά. Γίνεται πλέον κατανοητός ο τρόπος που η εφαρμογή επιτυγχάνει τους στόχους που θέσαμε στο πρώτο κεφάλαιο. Στο επόμενο κεφάλαιο θα δούμε κάποια συμπεράσματα και αναλυτικά δεδομένα για την εφαρμογή μας ενώ θα προτείνουμε και μερικές πιθανές μελλοντικές επεκτάσεις.

Κεφάλαιο 5 - Επίλογος

Στο κεφάλαιο αυτό θα δούμε τα συμπεράσματα που προκύπτουν αλλά και τις πιθανές μελλοντικές επεκτάσεις που θεωρούμε ότι θα μπορούσαν να αναπτύξουν την εφαρμογή μας θέτοντας νέους στόχους. Αρχικά γίνεται μία ανασκόπηση στα βασικά στοιχεία της εφαρμογής μας όπου συνοψίζουμε τα σημαντικότερα σημεία που αναφέραμε στο υπόλοιπο βιβλίο. Έπειτα κάνουμε την ανάλυση του κώδικα που συνθέτει την εφαρμογή μας, δίνοντας κάποιες μετρικές και εξηγώντας την σημασία τους. Στην ολοκλήρωση των συμπερασμάτων, βλέπουμε την εφαρμογή μας ως σύνολο και μέσο της ανάλυσης S.W.O.T. (Strengths, Weaknesses, Opportunities, Threats - Δυνατά σημεία, Αδυναμίες, Ευκαιρίες, Απειλές) καθορίζουμε τα δυνατότερα και πιο αδύναμα στοιχεία της τόσο εσωτερικά όσο και εξωτερικά της εφαρμογής. Τέλος, αναφέρουμε νέους πιθανούς στόχους που θα μπορούσαν να οδηγήσουν σε περεταίρω ανάπτυξη της εφαρμογής μας.

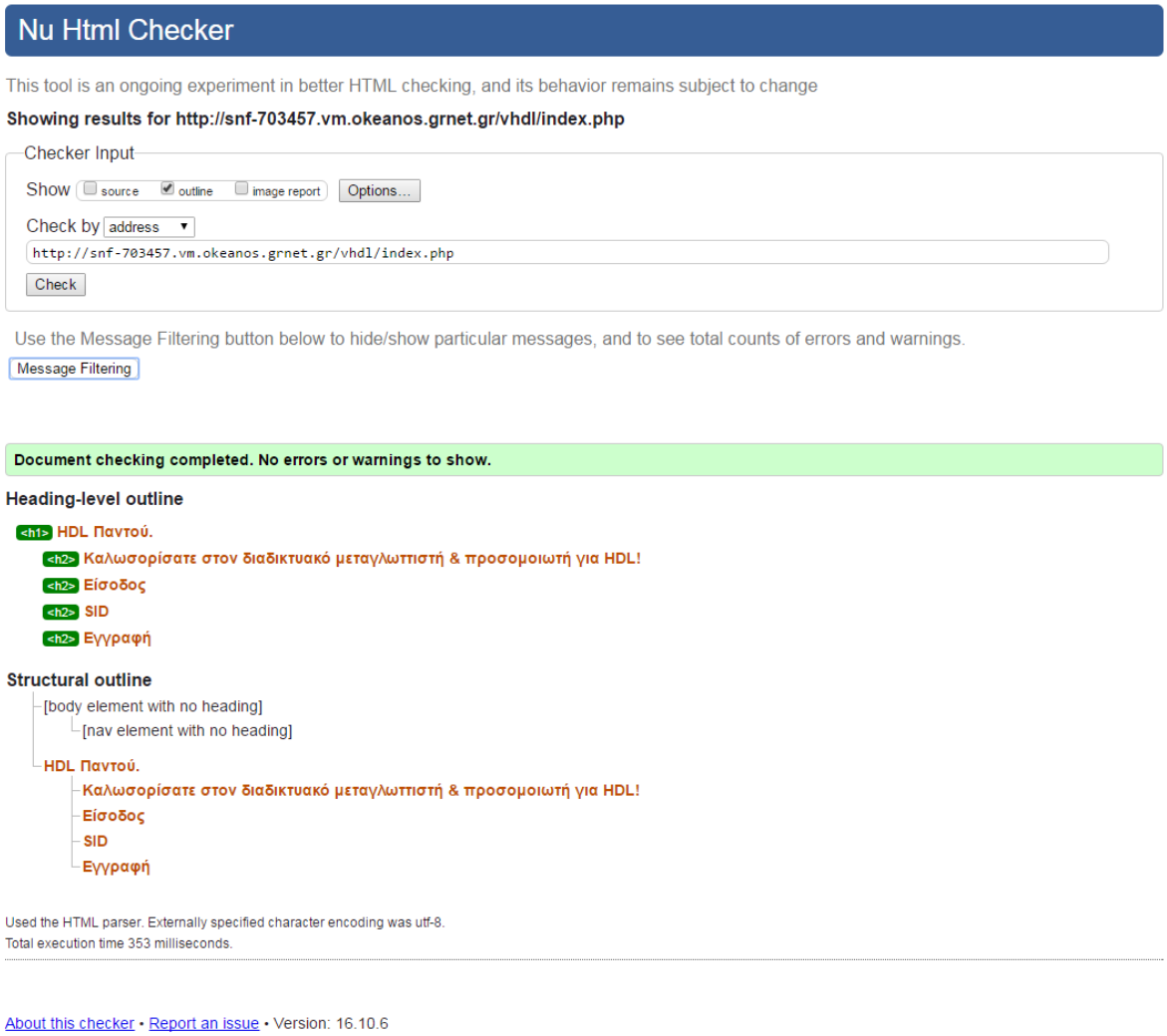
5.1 Συμπεράσματα

Στα προηγούμενα κεφάλαια είδαμε την εφαρμογή μας από την αρχική σχεδίαση και την διευκρίνηση των στόχων της ως την ανάπτυξη και τελικά την λειτουργική της εφαρμογή. Αναφέραμε ότι ο βασικός στόχος της εφαρμογής μας είναι η προσφορά ενός διαδικτυακού περιβάλλοντος για την δημιουργία και διατήρηση VHDL προγραμμάτων τα οποία μπορούν να μεταγλωττιστούν και προσομοιωθούν ώστε να δώσουν την δυνατότητα στον προγραμματιστή να αναλύσει τον τρόπο λειτουργίας τους. Κάναμε μία ανασκόπηση στα εργαλεία που αποτελούν την εφαρμογή μας και επιτρέπουν την ανάπτυξή της σε ένα διαδικτυακό

περιβάλλον. Στην συνέχεια προχωρήσαμε στην ανάλυση του τρόπου ανάπτυξης της εφαρμογής όπου αρχικά παρουσιάστηκε η διάρθρωση της βάσης δεδομένων και των καταλόγων αρχείων ενώ ολοκληρώθηκε με την λεπτομερή επεξήγηση των τεχνικών που χρησιμοποιήσαμε για την δημιουργία τόσο της εφαρμογής όσο και του εξωτερικού εργαλείου σχεδίασης κυματομορφών. Τέλος, είδαμε τις λειτουργίες που προσφέρει η εφαρμογή και τις περιπτώσεις χρήσης που αντιστοιχούν σε αυτές. Έτσι, έχουμε πλέον μία ολοκληρωμένη εικόνα για το σύνολο της εφαρμογής που παρουσιάζουμε και μπορούμε να εξάγουμε κάποια συμπεράσματα.

5.1.1 Ανάλυση Μετρικών Κώδικα

Σε αυτό το υποκεφάλαιο θα εστιάσουμε στην εφαρμογή μας ως ένα ολοκληρωμένο πρόγραμμα. Ως ένα ολοκληρωμένο πρόγραμμα διαδικτυακού προγραμματισμού, η εφαρμογή μας αποτελείται από μία ποικιλία γλωσσών. Ο κώδικα της κάθε γλώσσας μπορεί να αναλυθεί με την μέτρηση κάποιων μετρικών και την εξαγωγή συμπερασμάτων από αυτές. Αρχικά μπορούμε να δούμε ότι η εφαρμογή λειτουργεί αποτελεσματικά και δεν υπάρχει προβληματικός κώδικας, καθώς κάθε τέτοιο σημείο κώδικα όπου εντοπίστηκε έχει διορθωθεί.



Εικόνα 22: Επιβεβαίωση σωστής σχεδίας της εφαρμογής για το πρότυπο HTML5 από το εργαλείο που προσφέρει η W3C

Αυτό επιβεβαιώνεται και έως ένα ποσοστό από την επαλήθευση της εφαρμογής ως σελίδα HTML5 από το εργαλείο επαλήθευσης που προσφέρει ο οργανισμός 'World Wide Web Consortium' (W3C) όπως φαίνεται στην Εικόνα 22. Έτσι, οι πρώτες μετρικές που πρέπει να αναφέρουμε είναι αυτές που περιέχει ο Πίνακας 1. Ο πίνακας αυτός περιέχει τα αποτελέσματα ανάλυσης της εφαρμογής μας από το πρόγραμμα ανάλυσης προγραμμάτων CLOC [29]. Φαίνεται λοιπόν στο πίνακα ότι αναγνωρίζονται 4 γλώσσες προγραμματισμού στην εφαρμογή μας. Κυρίαρχο ρόλο έχει αναπάντεχα η γλώσσα CSS. Αυτό οφείλεται στην χρήση του πλαισίου Bootstrap που προσθέτει μεγάλο αριθμό αρχείων και γραμμών για τις εσωτερικές του λειτουργίες. Σε μικρότερο βαθμό, το ίδιο φαινόμενο παρατηρούμε στην γλώσσα JavaScript όπου εξωτερικά εργαλεία προσθέτουν μεγάλο μέρος του κώδικα. Η δική μας ανάπτυξη όμως επικεντρώνεται στην γλώσσα PHP.

Γλώσσα	**Αρχεία**	**Κενές γραμμές**	**Σχόλια**	**Κώδικας**
CSS	5	82	258	7618
PHP	50	386	272	4874
JavaScript	8	865	340	3263
Shell	1	21	22	59
Σύνολο	64	1354	892	15814

Πίνακας 1: Μετρικές αρχείων και κώδικα, αποτελέσματα του προγράμματος CLOC

Γι' αυτό βλέπουμε τις μετρικές κώδικα μόνο για PHP που περιέχει ο Πίνακας 2. Οι μετρικές του πίνακα είναι αποτελέσματα ανάλυσης της εφαρμογής από το πρόγραμμα PHPLOC [30]. Παρατηρούμε από τον πίνακα ότι ένα μικρό μέρος του κώδικα αποτελείται από σχόλια, όπως είναι άλλωστε αναμενόμενο. Καθώς χρησιμοποιούμε σχόλια για την γενική περιγραφή μίας μεθόδου και την επεξήγηση κομματιών κώδικα που εκτελούν μία λειτουργία, περιμένουμε τον αριθμό σχολιασμένων γραμμών να είναι περίπου όσες και οι ξεχωριστές λειτουργίες και υπολειτουργίες.

Σύνολο γραμμών κώδικα (LOC)	5146	
Σχολιασμένες γραμμές (CLOC)	272	(5.29%)
Γραμμές κώδικα (NCLOC)	4874	(94.71%)
Λογικές γραμμές κώδικα (LLOC)	1641	(31.89%)
Κώδικας σε κλάσεις	460	(28.03 %)
Μέσο μέγεθος κλάσης	115	
Μέσο μέγεθος μεθόδου	5	
Κώδικας εκτός κλάσεων	1181	(71.97 %)

Πίνακας 2: Βασικές μετρικές κώδικα PHP, αποτελέσματα του προγράμματος PHPLOC

Παρακάτω βλέπουμε την ανάλυση λογικών γραμμών κώδικα. Η μετρική αυτή υποδεικνύει τον αριθμό γραμμών εκτελέσιμου κώδικα αφαιρώντας το κλείσιμο και άνοιγμα αγκύλης, τις κενές γραμμές, τα σχόλια και ότι άλλο δεν εκτελείται άμεσα. Αρχικά από το γεγονός ότι οι λογικές γραμμές κώδικα είναι περίπου το ένα τρίτο του συνολικού κώδικα συμπεράνουμε ότι ο κώδικας είναι γραμμένος με σκοπό την εύκολη ανάγνωσή του από τον άνθρωπο αντί για την σύμπτυξή του για την ελαχιστοποίηση του μεγέθους των αρχείων. Αυτό προκύπτει από τον στόχο μας ως προς την ανάπτυξη μίας εύκολα συντηρήσιμης και επεκτάσιμης εφαρμογής. Έπειτα παρατηρούμε ότι από τις λογικές γραμμές κώδικα, ένα μεγάλο μέρος βρίσκεται μέσα σε

κλάσεις αλλά ο μέσος αριθμός γραμμών ανά μέθοδο είναι πολύ μικρός. Αυτή η μέτρηση είναι αποτέλεσμα της χρήσης πολλών μεθόδων για μικρές και συγκεκριμένες λειτουργίες που προσφέρουν μεγαλύτερη συνοχή κώδικα.

Μία άλλη σημαντική μετρική κώδικα είναι η κυκλωματική πολυπλοκότητα. Το πρόγραμμα PHPLOC δεν μας προσφέρει την μετρική αυτή απευθείας και άλλα παρόμοια προγράμματα δεν την υποστηρίζουν καθόλου λόγο της δυσκολίας απόκτησης σωστής μέτρησης. Παρόλα αυτά ο Πίνακας 3 περιέχει κάποιες σχετικές μετρικές από τα αποτελέσματα του προγράμματος PHPLOC. Παρατηρούμε από τον πίνακα ότι η μέση πολυπλοκότητα ανά λογική γραμμή κώδικα είναι αρκετά μικρή.

Μέση πολυπλοκότητα ανά LLOC	0.34
Μέση πολυπλοκότητα ανά κλάση	16.50
Ελάχιστη πολυπλοκότητα κλάσης	2.00
Μέγιστη πολυπλοκότητα κλάσης	34.00
Αριθμός κλάσεων	4
Μέση πολυπλοκότητα ανά μέθοδο	1.66
Ελάχιστη πολυπλοκότητα μεθόδου	1.00
Μέγιστη πολυπλοκότητα μεθόδου	6.00
Αριθμός μεθόδων	94

Πίνακας 3: Κυκλωματική πολυπλοκότητα κώδικα PHP, αποτελέσματα του προγράμματος PHPLOC

Μεγάλη αύξηση παρουσιάζεται στην πολυπλοκότητα κάποιων κλάσεων χωρίς όμως να αντικατοπτρίζεται εσωτερικά στις μεθόδους. Παίρνοντας υπόψη την μέγιστη πολυπλοκότητα κλάσης που εμφανίζει μία ακραία τιμή αντιλαμβανόμαστε ότι υπάρχουν συγκεκριμένα κομμάτια κώδικα με μεγάλη πολυπλοκότητα που ανεβάζουν σημαντικά τον μέσο όρο. Τέτοια κομμάτια κώδικα εμφανίζονται σε μεθόδους όπως αυτή που εξάγει τα περιεχόμενα ενός συμπιεσμένου αρχείου. Σε κομμάτια κώδικα όπως αυτό πρέπει να γίνουν υποχρεωτικά εμφωλευμένοι έλεγχοι και επαναλήψεις ώστε να εκτελέσουν αποτελεσματικά την λειτουργία τους. Ακόμη όμως και με κομμάτια κώδικα αυτού του είδους, η εφαρμογή μας έχει ως σύνολο σχετικά μικρή πολυπλοκότητα για τις λειτουργίες που εκτελεί. Συμπεράνουμε λοιπόν από τις παραπάνω μετρικές ότι έχουμε καταφέρει την δημιουργία μίας μεγάλης εφαρμογής με πολλές λειτουργίες που όμως απαρτίζονται από εύκολα συντηρήσιμο και επεκτάσιμο κώδικα.

5.1.2 Ανάλυση S.W.O.T.

Η ανάλυση S.W.O.T. αντιμετωπίζει την εφαρμογή μας ως ένα καινοτόμο έργο λογισμικού και επικεντρώνεται στα δυνατά και αδύναμα στοιχεία του αλλά και τις ευκαιρίες και απειλές που μπορεί να αντιμετωπίσει. Για την σωστή ανάλυση δίνουμε αντικειμενικά τα σημαντικότερα στοιχεία της εφαρμογής μας σε κάθε κατηγορία.

Δυνατά Σημεία	**Αδυναμίες**
• Είναι εφαρμογή ανοιχτού κώδικα. • Διανέμεται δωρεάν. • Υποστηρίζει ποικιλία συσκευών. • Εύκολα συντηρήσιμη και επεκτάσιμη. • Δεν απαιτεί την εγκατάσταση προγραμμάτων στο σύστημα του χρήστη. • Χρησιμοποιεί εύχρηστο και φιλικό προς στον χρήστη περιβάλλον. • Δυνατότητες δοκιμής των λειτουργιών από τους επισκέπτες.	• Απαιτεί εξοικείωση με διαδικτυακό εξυπηρετητή για την εγκατάσταση. • Κάποιες λειτουργίες της εφαρμογής απαιτούν ενεργή διαχείριση. • Η μεταγλώττιση και προσομοίωση γίνεται με καθυστέρηση λόγο της χρήσης εξωτερικού προγράμματος.
Ευκαιρίες	**Απειλές**
• Ανάπτυξη της GHDL για την υποστήριξη νέων γλωσσών HDL. • Ανάπτυξη συμβατών προγραμμάτων που μπορεί να επιτρέψουν και την λειτουργία της σύνθεσης ενός έργου σε πλακέτα. • Αναγνώριση της εφαρμογής από μεγαλύτερο κοινό και η συνεισφορά τους στην περαιτέρω ανάπτυξή της.	• Έλλειψη ενδιαφερόμενου κοινού ώστε να δικαιολογηθεί η χρήση πόρων του εξυπηρετητή. • Αντικατάσταση της γλώσσας VHDL από νέα.

Πίνακας 4: Ανάλυση S.W.O.T. της εφαρμογής

Έτσι λοιπόν ο Πίνακας 4 περιέχει την ανάλυση που κάναμε. Εσωτερικά της εφαρμογής μας, είναι φανερό ότι τα δυνατά της σημεία προκύπτουν από τον σωστό σχεδιασμό και επιλογή κατάλληλων εργαλείων για την ανάπτυξή της αλλά και από τις ιδιότητές της ως διαδικτυακή εφαρμογή που διανέμεται δωρεάν και είναι ανοιχτού κώδικα. Οι αδυναμίες της βρίσκονται κυρίως στην χρήση περιορισμένων εξωτερικών εργαλείων. Παρόλο που αυτά τα εργαλεία είναι τα καλύτερα δυνατά για την επίτευξη των στόχων μας, παραμένουν ένα κέντρο αδυναμίας. Έτσι βλέπουμε ότι οι πρώτες ευκαιρίες που αναφέρονται επικεντρώνονται στην ανάπτυξη ή αντικατάσταση αυτών των προγραμμάτων. Παράλληλα φυσικά έχουμε και την δημιουργία ενός μεγαλύτερου κοινού για την εφαρμογή μας καθώς η χρήση της θα οδηγήσει στην βελτίωσή της. Τέλος, οι απειλές προκύπτουν από την έλλειψη χρήσης της εφαρμογής μας. Η έλλειψη αυτή μπορεί να οδηγήσει στην απόσυρση της εφαρμογής μας είτε με άμεσο τρόπο λόγο του μικρού ενδιαφερόμενου κοινού ή με έμμεσο λόγο της αντικατάστασης της γλώσσας VHDL από νεότερη. Έχοντας ολοκληρώσει την ανάλυση S.W.O.T. έχουμε μία πιο ξεκάθαρη εικόνα της εφαρμογής μας και του τρόπου που μπορεί να ανταπεξέλθει σε ένα ανταγωνιστικό περιβάλλον. Στο επόμενο υποκεφάλαιο θα παρουσιάσουμε κάποιες προτάσεις που αντιμετωπίζουν τις αδυναμίες και προσπαθούν να δημιουργήσουν νέες ευκαιρίες για την εφαρμογή μας.

5.2 Μελλοντικές Επεκτάσεις

Έχοντας πλέον μία πλήρη εικόνα της εφαρμογής μας, μπορούμε να προβλέψουμε κάποιους νέους στόχους και να προτείνουμε μελλοντικές επεκτάσεις για την υλοποίησή τους. Κάποιες επεκτάσεις έχουν ήδη γίνει από την αρχική ιδέα της εφαρμογής. Τέτοιες επεκτάσεις είναι οι λειτουργίες των εξαρτημάτων, η εισαγωγή άλλων χρηστών ως συντάκτες σε κάποιο έργο και η υποστήριξη της ελληνικής αλλά και αγγλικής γλώσσας. Οι επεκτάσεις όμως που προτείνουμε παρακάτω ξεφεύγουν από τα όρια αυτής της εργασίας και κάποιες από αυτές απαιτούν αρκετή μελέτη και χρόνο ώστε να δικαιολογούν ξεχωριστές διπλωματικές εργασίες.

Η πρώτη επέκταση που προτείνουμε προκύπτει από τις αδυναμίες τις εφαρμογής που αναφέραμε στο προηγούμενο υποκεφάλαιο. Η επέκταση αυτή αποσκοπεί στην ανάπτυξη ενός εσωτερικού εργαλείου μεταγλώττισης και προσομοίωσης. Με την επέκταση αυτή οι λειτουργίες μεταγλώττισης και προσομοίωσης είναι πλέον κομμάτι της εφαρμογής μας η ταχύτητα με την οποία ολοκληρώνονται οι διαδικασίες τους είναι σημαντικά μεγαλύτερη. Φυσικά η ανάπτυξη ενός προσομοιωτή και πιθανός ενός μεταγλωττιστή είναι ένα μεγάλο έργο που απαιτεί εξοικείωση τόσο με τον τρόπο λειτουργία τους όσο και με την γλώσσα την οποία προσομοιώνει και μεταγλωττίζει. Η επέκταση αυτή δεν είναι μία απλή βελτίωση. Με την σωστή της εφαρμογή μπορεί όχι απλώς να οδηγήσει στην απλοποίηση των διαδικασιών μεταγλώττισης και προσομοίωσης αλλά και στην πιθανή εισαγωγή νέων στόχων. Τέτοιοι στόχοι μπορεί να αποτελούνται από την υποστήριξη επιπλέον γλωσσών HDL ή ακόμη και από την προσθήκη της λειτουργίας σύνθεσης του έργου σε πλακέτα. Γίνεται εύκολα αντιληπτό ότι η επέκταση που προτείνουμε ανοίγει νέες δυνατότητες αλλά ταυτόχρονα απαιτεί μεγάλη επένδυση χρόνου και προσπάθειας για την επίτευξή τους.

Η δεύτερη επέκταση που προτείνουμε είναι η υποστήριξη επιπλέον γλωσσών HDL. Είδαμε ήδη ότι αυτή η επέκταση μπορεί να αποτελέσει συνέχεια της πρώτης. Είναι όμως δυνατή και με την χρήση εξωτερικών προγραμμάτων με παρόμοιο τρόπο που χρησιμοποιούμε το GHDL για την VHDL. Σε αυτή την περίπτωση θα χρειαστεί φυσικά περεταίρω έρευνα για τα διαθέσιμα προγράμματα, την συμβατότητά τους με το σύστημα που χρησιμοποιούμε και τους πιθανούς τρόπους διασύνδεσής τους με την εφαρμογή μας. Πολλοί προσομοιωτές δεν έχουν σωστή και πλήρη τεκμηρίωση οπότε η έρευνα αυτή πρέπει σε πολλές περιπτώσεις να γίνει πειραματικά. Άλλοι προσομοιωτές που καλύπτουν τις ανάγκες συμβατότητας είναι εμπορικής φύσεως και θα πρέπει η χρήση τους να γίνει μέσο συμφωνίας με τον προμηθευτή. Βλέπουμε ότι η επέκταση αυτή μπορεί είτε να αποτελέσει μέρος ενός μεγάλου έργου είτε να είναι αποτέλεσμα διεξοδικής έρευνας και πιθανών συμφωνιών με εξωτερικούς φορείς. Σε κάθε περίπτωση όμως η επέκταση αυτή προσφέρει μια σημαντική αύξηση προστασίας από τις πιθανές απειλές για την εφαρμογή μας.

Τέλος, προτείνουμε μία επέκταση που έχει αρκετά διαφορετικό στόχο. Η επέκταση αυτή αποτελεί συνηθισμένη πρακτική σε διαδικτυακές εφαρμογές που προσφέρουν την μεταγλώττιση, προσομοίωση ή εκτέλεση κάποιας γλώσσας προγραμματισμού. Πρόκειται για την εισαγωγή οδηγιών προγραμματισμού στην γλώσσα VHDL. Ο στόχος μίας τέτοιας επέκτασης παίρνει μία διαφορετική κατεύθυνση από αυτούς που έχουμε θέσει και αποσκοπεί στην εκπαίδευση και βελτιστοποίηση του προγραμματιστή. Η υλοποίηση αυτής της επέκτασης είναι μία σχετικά απλή διαδικασία αλλά η συγγραφή και σωστή απεικόνιση του εκπαιδευτικού υλικού απαιτεί ως συνήθως σημαντικό χρόνο και προσπάθεια. Δεδομένου όμως ότι η εφαρμογή

αυτή αναπτύχθηκε ως διπλωματική εργασία σε ένα εκπαιδευτικό ίδρυμα, είναι πολύ πιθανό μία επέκταση με σκοπό την εκπαίδευση να έχει προτεραιότητα.

Βλέποντας λοιπόν πιθανά μελλοντικά βήματα της εφαρμογής μας, κλείνουμε την ανάλυσή της. Η εφαρμογή που αναπτύξαμε προσφέρει ένα ολοκληρωμένο περιβάλλον για την μεταγλώττιση και προσομοίωση ενός VHDL έργου. Προσφέρει λειτουργίες που όχι μόνο καλύπτουν τους στόχους της αλλά αγγίζουν τα όρια νέων στόχων. Όπως είδαμε στο τελευταίο κεφάλαιο, η εισαγωγή νέων στόχων είναι δυνατή με την επένδυση αρκετού χρόνου και προσπάθειας, η υλοποίησή τους όμως μπορεί να αποτελέσει το μέλλον της εφαρμογής μας.

Βιβλιογραφία

[1] «ModelSim ASIC and FPGA Design - Mentor Graphics,» [Ηλεκτρονικό]. Available: https://www.mentor.com/products/fv/modelsim/. [Πρόσβαση Αύγουστος 2016].

[2] I. S. 1666-2005, «IEEE Standard System C Language Reference Manual».

[3] «Vivado Design Suite,» [Ηλεκτρονικό]. Available: http://www.xilinx.com/products/design-tools/vivado.html. [Πρόσβαση Αύγουστος 2016].

[4] «ISE Simulator (ISim),» [Ηλεκτρονικό]. Available: http://www.xilinx.com/products/design-tools/isim.html. [Πρόσβαση Αύγουστος 2016].

[5] F. Tom, «Vivado design suite,» White Paper 5, 2012.

[6] «Icarus Verilog,» [Ηλεκτρονικό]. Available: http://iverilog.icarus.com/. [Πρόσβαση Αύγουστος 2016].

[7] «GTKWave,» [Ηλεκτρονικό]. Available: http://gtkwave.sourceforge.net/. [Πρόσβαση Αύγουστος 2016].

[8] «Yosys Open Synthesis Suite,» [Ηλεκτρονικό]. Available: http://www.clifford.at/yosys/. [Πρόσβαση Αύγουστος 2016].

[9] «Verilog Online Simulator,» [Ηλεκτρονικό]. Available: http://www.iverilog.com/index.php. [Πρόσβαση Αύγουστος 2016].

[10] «EDA Playground,» [Ηλεκτρονικό]. Available: http://www.edaplayground.com/. [Πρόσβαση Αύγουστος 2016].

[11] «HTML5,» W3C, [Ηλεκτρονικό]. Available: https://www.w3.org/TR/2014/REC-html5-20141028/. [Πρόσβαση Οκτώβριος 2016].

[12] S. Pieters και A. van Kesteren, *HTML5 Differences from HTML4*, World Wide Web Consortium, 2014.

[13] «CSS Snapshot 2015,» W3C, [Ηλεκτρονικό]. Available: https://www.w3.org/TR/2015/NOTE-css-2015-20151013/. [Πρόσβαση Οκτώβριος 2016].

[14] «ECMAScript 2016 Language Specification,» ECMA International, [Ηλεκτρονικό]. Available: https://tc39.github.io/ecma262/2016/. [Πρόσβαση Οκτώβριος 2016].

[15] D. Flanagan, σε *JavaScript: the definitive guide*, O'Reilly Media, Inc., 2006, pp. 381-404.

[16] «PHP: Language Reference - Manual,» The PHP Group, [Ηλεκτρονικό]. Available: http://php.net/manual/en/langref.php. [Πρόσβαση Οκτώβριος 2016].

[17] «PHP: History of PHP,» [Ηλεκτρονικό]. Available: https://secure.php.net/manual/en/history.php.php. [Πρόσβαση Αύγουστος 2016].

[18] «MySQL Documentation,» Oracle Corporation, [Ηλεκτρονικό]. Available: http://dev.mysql.com/doc/. [Πρόσβαση Οκτώβριος 2016].

[19] W. Cao, F. Yu και J. Xie, «Realization of the low cost and high performance mysql cloud database.,» σε *Proceedings of the VLDB Endowment 7.13*, 2014.

[20] «PHP: Introduction,» [Ηλεκτρονικό]. Available: http://php.net/manual/en/intro.pdo.php. [Πρόσβαση Αύγουστος 2016].

[21] R. T. Fielding, J. Gettys, J. C. Mogul, H. F. Nielsen, L. Masinter, P. J. Leach και T. Berners-Lee, « Hypertext Transfer Protocol -- HTTP/1.1,» The Internet Society, 1999.

[22] «Bootstrap · The world's most popular mobile-first and responsive front-end framework.,» [Ηλεκτρονικό]. Available: http://getbootstrap.com/. [Πρόσβαση Αύγουστος 2016].

[23] «jQuery,» [Ηλεκτρονικό]. Available: https://jquery.com/. [Πρόσβαση Αύγουστος 2016].

[24] «Ace - The High Performance Code Editor for the Web,» [Ηλεκτρονικό]. Available: https://ace.c9.io. [Πρόσβαση Αύγουστος 2016].

[25] «GHDL Main/Home Page,» [Ηλεκτρονικό]. Available: http://ghdl.free.fr/. [Πρόσβαση Αύγουστος 2016].

[26] R. L. Rivest, «The MD5 Message-Digest Algorithm,» [Ηλεκτρονικό]. Available: https://www.ietf.org/rfc/rfc1321.txt. [Πρόσβαση Οκτώβριος 2016].

[27] I. S. 1364-2001, «IEEE Standard Verilog Hardware Description Language».

[28] «JSON,» [Ηλεκτρονικό]. Available: http://www.json.org/. [Πρόσβαση Οκτώβριος 2016].

[29] «CLOC -- Count Lines of Code,» Al Danial, [Ηλεκτρονικό]. Available: http://cloc.sourceforge.net/. [Πρόσβαση Οκτώβριος 2016].

[30] «PHPLOC,» Sebastian Bergmann, [Ηλεκτρονικό]. Available: https://github.com/sebastianbergmann/phploc. [Πρόσβαση Οκτώβριος 2016].