ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

ΣΧΟΛΗΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Υλοποίηση διαδικτυακής εφαρμογής τρισδιάστατής διαδραστικής απεικόνισης των μνημείων της Κρήτης με χρήση WebGL

Κοτσαρίνης Παναγιώτης

Εξεταστική Επιτροπή

Κατερίνα Μανιά (ΗΜΜΥ)

Σταύρος Χριστοδουλάκης(ΗΜΜΥ)

Παναγιώτης Παρθένιος(ΑρΜΗΧ)

Technical University of Crete

School of Electronic and Computer Engineering

Implementation of a 3D Web Application for the interactive visualization of cultural Monuments of Crete using WebGL

Kotsarinis Panagiotis

Thesis Committee

Katerina Mania (ECE)

Stavros Christodoulakis(ECE)

Panagiotis Parthenios(ARCH)

Περίληψη

Στα πλαίσια της συγκεκριμένης διπλωματικής εργασίας αναπτύχθηκε μία διαδραστική τρισδιάστατη διαδικτυακή εφαρμογή αναπαράστασης των μνημείων της Κρήτης. Η εφαρμογή αυτή αναπαριστά δεκαπέντε επιλεγμένα μνημεία της Κρήτης σε πέντε διαφορετικά επίπεδα λεπτομέρειας και παρέχει στον χρήστη την δυνατότητα να παρακολουθήσει την εξέλιξη του κάθε μνημείου σε εφτά διαφορετικές ιστορικές περιόδους. Τα επίπεδα λεπτομέρειας είναι τα Κρήτη, Νομός, Οικισμός, Κτηριακό Συγκρότημα και Μνημείο και ξεκινούν από σημειακή αναφορά των μνημείων και φτάνουν σε απεικόνιση του μνημείου αφαιρετικά τόσο όσο να διακρίνεται από την μοναδικότητα του. Η απεικόνιση σε διαφορετικές ιστορικές περιόδους αφορά τις εξής: Μοντέρνα, Οθωμανική, Ενετική, Βυζαντινή, Ρωμαϊκή, Ελληνιστική και Μινωική εποχή. Στο επίπεδο Μνημείο κάθε μνημείου, υπάρχουν επιπλέον πληροφορίες που το αφορούν, Ιστορικά στοιχεία, Βίντεο και Φωτογραφίες σχετικά με αυτό, Χάρτης, Επιπλέον πληροφορίες καθώς και σχετικοί εξωτερικοί σύνδεσμοι. Επιπλέον αναπτύχθηκε ένα διαδικτυακό εργαλείο διαχείρισης της εφαρμογής αυτής το οποίο υλοποιήθηκε για την ανάπτυξη της. Για την τρισδιάστατη αναπαράσταση των τρισδιάστατων μοντέλων έγινε χρήση της WebGL, τεχνολογίας που επιτρέπει την απεικόνιση τρισδιάστατων σκηνών σε συμβατούς φυλλομετρητές. Σχεδιάστηκε και υλοποιήθηκε η απαραίτητη βάση δεδομένων για την αποθήκευση των δεδομένων έτσι ώστε να υποστηρίζεται η επεκτασιμότητα της εφαρμογής. Τέλος αναπτύχθηκε ο απαραίτητος κώδικας στην μεριά του διακομιστή για την επικοινωνία της βάσης δεδομένων με τον φυλλομετρητή μέσω υπηρεσιών που υλοποιήσαμε.

Abstract

This diploma thesis presents a 3D Web Application for the interactive visualization of cultural Monuments of Crete. The implementation of the web-based application is based on WebGL. The application visualizes each cultural monument in five spatial levels of detail representing initially Crete as a whole, then by prefecture, region, complex of monuments and finally focusing on the actual monument. Simultaneously, each level of detail is visualized in seven 7 different time periods. The user is able to virtually visit Crete across regions and time. The user interface consists of two bars, one vertical and one horizontal representing the level of detail and the time periods respectively. The user can click on the desired level of detail and historical period in order to view in 3D the appropriate representation by simple interaction with the mouse. The users can also navigate inside the 3D models by performing simple mouse events interactively. When the user selects the last level of detail of the spatial axis visualizing an interactive monument, a menu is appearing offering certain options. At the right side of the screen an arrow appears and when the user slides it, a slide menu is available including monument information. The user can select photos, videos, historical and general information associated to each monument. We designed a database containing the 3D models and their associated information using Ajax technologies enabling the asynchronous loading of suitable 3D models without reloading the page. Finally, we implemented a management tool for the administrator of the main application.

Θα ήθελα να ευχαριστήσω την κ.Κατερίνα Μανιά η οποία μου εμπιστεύτηκε την υλοποίηση της εφαρμογής αυτής, η οποία ως παραδοτέο έργο έπρεπε να έχει επιτυχή έκβαση.

Επιπλέον θα ήθελα να ευχαριστήσω τον κ. Παναγιώτη Παρθένιο ο οποίος με εμπιστεύτηκε και αυτός για την υλοποίηση της εφαρμογής και είχαμε μία επιτυχημένη συνεργασία.

Την ομάδα του παραδοτέου έργου η οποία αποτελούνταν από φοιτητές και καθηγητές της αρχιτεκτονικής σχολής για την υλοποίηση των τρισδιάστατων μοντέλων και την όμορφη συνεργασία μας.

Την Μαριάννα Δημητρίου η οποία συμμετείχε στην ομάδα ως συμβουλευτικό και οργανωτικό μέλος.

Τέλος ένα μεγάλο και ιδιαίτερο ευχαριστώ στην οικογένεια μου για την απόλυτη στήριξή τους σε αγχωτικές περιόδους της υλοποίησης καθώς επίσης και τους φίλους που με στήριξαν.

Περιεχόμενα

Πίνακας εικόνων

1 Εισαγωγή

1.1 Εισαγωγή

Διανύουμε μία εποχή όπου η εξέλιξη της τεχνολογίας τόσο από άποψη υλικού, όσο και λογισμικού έχει κάνει την σχεδίαση, την προβολή ακόμη και την εκτύπωση τρισδιάστατων αντικειμένων προσβάσιμη στον καθένα. Τρισδιάστατα γραφικά βρίσκουμε και στο διαδίκτυο κυρίως με την χρήση κάποιου επιπλέον προγράμματος (plugin) στον φυλλομετρητή που χρησιμοποιούμε. Η ιδέα της διαδραστικής απεικόνισης τρισδιάστατων μοντέλων μέσω φυλλομετρητή είχε κάνει την εμφάνιση της από το 1994 με την πρώτη έκδοση της VRML(Virtual Reality Markup Language), η οποία με την χρήση plugin έκανε εφικτό τον στόχο αυτό. Όμως οι υπολογιστές της εποχής με την χαμηλή υπολογιστική τους ισχύ και οι αργές ταχύτητες του διαδικτύου δεν βοήθησαν στην εξέλιξη της τρισδιάστατης απεικόνισης στο διαδίκτυο. Με την χρήση επιπρόσθετων προγραμμάτων(Adobe Flash Player, Silverlight, Shockwave Player κ.α) στον φυλλομετρητή επανήλθε ο τρισδιάστατος κόσμος στο διαδίκτυο , κυρίως σε διαδικτυακά παιχνίδια. Η εμφάνιση της WebGL το 2011 έφερε τον τρισδιάστατο κόσμο στον φυλλομετρητή, χωρίς την χρήση plugin. Ουσιαστικά παρέχει στον browser πρόσβαση στην κάρτα γραφικών του υπολογιστή, για τους πολύπλοκους υπολογισμούς που χρειάζονται τα τρισδιάστατα γραφικά. Σε συνδυασμό με τις πολύ γρήγορες ταχύτητες του διαδικτύου και τις πολύ υψηλές αποδόσεις καρτών γραφικών ακόμη και φτηνών υπολογιστών, αρχίζει το διαδίκτυο να υιοθετεί τρισδιάστατα γραφικά.

1.2 Στόχος της Διπλωματικής Εργασίας

Σκοπός της συγκεκριμένης διπλωματικής εργασίας είναι η ανάπτυξη μίας διαδικτυακής εφαρμογής τρισδιάστατης διαδραστικής απεικόνισης των μνημείων της Κρήτης καθώς και την εξέλιξή τους στον χρόνο χωρίς την χρήση επιμέρους προγράμματος (plugin) στον φυλλομετρητή (browser).

Η εφαρμογή αυτή έχει σαν κύριο στόχο την υλοποίηση μιας αμιγώς τρισδιάστατης εφαρμογής στο διαδίκτυο η οποία παρέχει διαδραστικότητα μέσω της τρισδιάστατης σκηνής που απεικονίζει. Πιο συγκεκριμένα παρουσιάζονται τα μνημεία της Κρήτης σε σημειακό επίπεδο, και μέσω της διαδραστικότητας του χρήστη παρουσιάζονται σαν τρισδιάστατα μοντέλα με περισσότερη λεπτομέρεια. Για να υλοποιηθεί η εφαρμογή αυτή και να μπορέσει να παραδοθεί έπρεπε να την υλοποιήσουμε με τέτοιο τρόπο ούτως ώστε να μεταφέρεται μικρός όγκος δεδομένων για να μειωθεί η αναμονή του χρήστη, αλλά και να μειωθούν οι απαιτήσεις σε υπολογιστικούς πόρους καθώς στόχος ήταν η επισκεψιμότητα της εφαρμογής από ένα μέσο ηλεκτρονικό υπολογιστή με μέση ταχύτητα σύνδεσης στο διαδίκτυο.

Η ιδέα προήλθε από τον κ.Παρθένιο, επίκουρος καθηγητής της σχολής αρχιτεκτόνων μηχανικών του πολυτεχνείου Κρήτης, και χρηματοδοτήθηκε από την Cyta Hellas για την περάτωση της. Για την υλοποίηση της εφαρμογής αυτής συνεργαστήκαμε με φοιτητές και καθηγητές της αρχιτεκτονικής οι οποίοι σχεδίασαν και μας παρείχαν τα τρισδιάστατα μοντέλα και τις πληροφορίες γύρω από αυτά.Στην εφαρμογή αρχικός στόχος ήταν η εισαγωγή 15 μνημείων με πλήρη λεπτομέρεια, και 70 μνημείων σε σημειακή μορφή.

1.3 Σύντομη Περιγραφή

Αρχικά θα πρέπει να αναφέρουμε ότι για την επίτευξη του στόχου της διπλωματικής εργασίας, ουσιαστικά δημιουργήσαμε την βασική εφαρμογή για τους χρήστες και μία εφαρμογή διαχείρισης.

Στην εφαρμογή του διαχειριστή, μπορεί ο χρήστης να εισάγει το τρισδιάστατο μοντέλο του, και να επιλέξει αν αυτό το μοντέλο οδηγεί σε κάποιο άλλο μοντέλο εάν επιλεχθεί(on-click). Μέσω αυτής της εφαρμογής καταφέραμε να εισάγουμε όλα τα δεδομένα μας και την διαδραστικότητα στην εφαρμογή. Τέλος ,μέσω της εφαρμογής διαχείρισης μπορεί να εμπλουτιστεί στο μέλλον η βασική μας εφαρμογή με περεταίρω μνημεία.

Η εφαρμογή για τους απλούς χρήστες παρέχει την δυνατότητα στον χρήστη να δει τα επιλεγμένα 15 μνημεία της Κρήτης σε τρισδιάστατα μοντέλα, και την εξέλιξη τους στον χρόνο μέσω του φυλλομετρητή.

Υπάρχουν 5 επίπεδα λεπτομέρειας:

- Κρήτη
- Νομός
- Περιφέρεια
- Σύμπλεγμα Μνημείων
- Μνημείο

και 7 βασικές χρονικές περίοδοι που επιλέχθηκαν:

- Μινωική περίοδος
- Αρχαιοελληνική περίοδος
- Ρωμαϊκή περίοδος
- Βυζαντινή περίοδος
- Ενετική περίοδος
- Οθωμανική περίοδος
- Σύγχρονη εποχή

Ξεκινώντας από την Κρήτη επιλέγουμε κάποιο νομό ο οποίος έχει πινέζες με τα μνημεία του και εν συνεχεία επιλέγοντας κάποια πινέζα μεταβαίνουμε στην περιφέρεια του μνημείου από που πάλι με την χρήση της πινέζας προχωράμε στην ευρύτερη περιοχή του μνημείου και τέλος στο ίδιο το μνημείο. Στο τελευταίο επίπεδο λεπτομέρειας (Μνημείο) μπορούμε να δούμε πληροφορίες για το εκάστοτε μνημείο όπως μερικά ιστορικά στοιχεία, βίντεο και φωτογραφίες του μνημείου, χάρτης της περιοχής, πληροφορίες επισκεψιμότητας και εξωτερικούς συνδέσμους.

Σε όλα τα επίπεδα μπορούμε να δούμε την εξέλιξη του μνημείου στην περίοδο του χρόνου. Για να εντοπίζονται οι αλλαγές έχει επιλεγεί να είναι συμπαγές το υλικό του τρισδιάστατου μοντέλου την εποχή που χτίστηκε, και διάφανο στις μεταγενέστερες εποχές.

1.4 Δομή της Εργασίας

Σε αυτή την ενότητα, ολοκληρώνοντας το πρώτο κεφάλαιο θα παρουσιάσουμε τον τρόπο με τον οποίο έχει δομηθεί η διπλωματική αυτή εργασία και θα αναφέρουμε περιληπτικά το περιεχόμενο των κεφαλαίων που ακολουθούν.

Στο κεφάλαιο 2 παρουσιάζεται η σχετική έρευνα την οποία πραγματοποιήσαμε για την επιλογή των τεχνολογιών μέσω των οποίων υλοποιήθηκε η εφαρμογή την οποία περιγράφουμε. Η έρευνα αυτή πραγματοποιήθηκε πριν ξεκινήσει ο σχεδιασμός και η υλοποίηση της εφαρμογής.

Στο κεφάλαιο 3 παρουσιάζονται οι τεχνολογίες οι οποίες επιλέχθηκαν και χρησιμοποιήθηκαν για την υλοποίηση της παρούσας διπλωματικής, καθώς επίσης γίνεται και ανάλυση της χρήσης τους.

Στο κεφάλαιο 4 γίνεται περιγραφή της εφαρμογής η ανάλυση των απαιτήσεων της καθώς επίσης και ανάλυση των χρηστών στους οποίους απευθύνεται, ώστε να προκύψουν οι λειτουργικότητες οι οποίες πρέπει να υλοποιηθούν για την περάτωση της.

Το κεφάλαιο 5αφορά την υλοποίηση της εφαρμογής. Γίνεται παρουσίαση της δομής της όπως χωρίστηκε σε επίπεδα και παρουσιάζονται και επεξηγούνται τα πιο σημαντικά μέρη της υλοποίησης μέσω των τεχνολογιών που αναλύθηκαν στο τρίτο κεφάλαιο.

Στο κεφάλαιο 6 γίνεται παρουσίαση του γραφικού σχεδιασμού της εφαρμογής ο οποίος πραγματοποιήθηκε στα πρώτα στάδια του σχεδιασμού και της ανάλυσης των λειτουργιών της εφαρμογής. Επιπλέον παρουσιάζεται η γραφική διεπαφή χρήστη όπως αυτή υλοποιήθηκε και παρουσιάζεται και στους τελικούς χρήστες της εφαρμογής.

Στο κεφάλαιο 7 γίνεται ανασκόπηση της διπλωματικής εργασίας και αναφέρουμε και κάποιες πιθανές μελλοντικές επεκτάσεις οι οποίες θα μπορούσαν να διευρύνουν την λειτουργικότητα και την απήχηση της.

2 Επισκόπηση Σχετικής Έρευνας

2.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα αναφερθούμε στην έρευνα που πραγματοποιήθηκε για να διαπιστώσουμε αν είναι εφικτός ο στόχος της εφαρμογής που θέλαμε να υλοποιήσουμε. Θα αναφερθούμε λοιπόν σε παρόμοιες εφαρμογές που υπήρχαν, στις τεχνολογίες που μπορούσαν να μας εξυπηρετήσουν, και στα προβλήματα που είχαμε να αντιμετωπίσουμε σε σχέση με την τεχνολογία που είχαμε στην διάθεση μας. Τέλος θα αναφερθούμε στις επιλογές που κάναμε, και στις αποφάσεις που χρειάστηκαν να παρθούν.

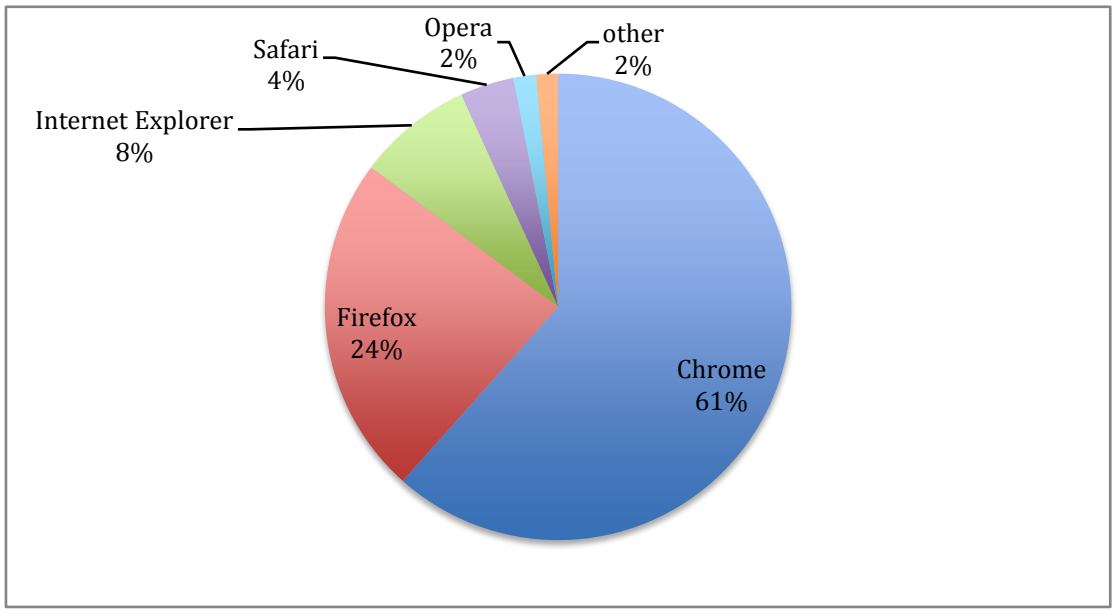
2.2 Γενικές απαιτήσεις Διαδικτυακών Εφαρμογών

Διαδικτυακές εφαρμογές είναι εκείνες, οι οποίες είναι προσβάσιμες μέσω του διαδικτύου και χρησιμοποιούν γλώσσες προγραμματισμού που υποστηρίζονται από φυλλομετρητές(όπως ο συνδυασμός της Javascript, HTML και CSS). Οι εφαρμογές αυτές εκτελούνται σε ισχυρές υπολογιστικές μηχανές οι οποίες έχουν τον ρόλο του σταθμού εξυπηρέτησης(web server) και παρέχουν τις υπηρεσίες τους σε περισσότερους του ενός χρήστη (client). Οι διαδικτυακές εφαρμογές ανάλογα με τις λειτουργίες που υποστηρίζουν χωρίζονται συνήθως σε τρείς βασικές βαθμίδες. Την βαθμίδα παρουσίασης, την βαθμίδα εκτέλεσης και την βαθμίδα αποθήκευσης. Ο φυλλομετρητής αποτελεί την βαθμίδα παρουσίασης, ο διακομιστής ιστού που χρησιμοποιεί μία από τις τεχνολογίες υποστήριξης δυναμικού διαδικτυακού περιεχομένου αποτελεί την βαθμίδα εκτέλεσης και τέλος η βάση δεδομένων αποτελεί την βαθμίδα αποθήκευσης.

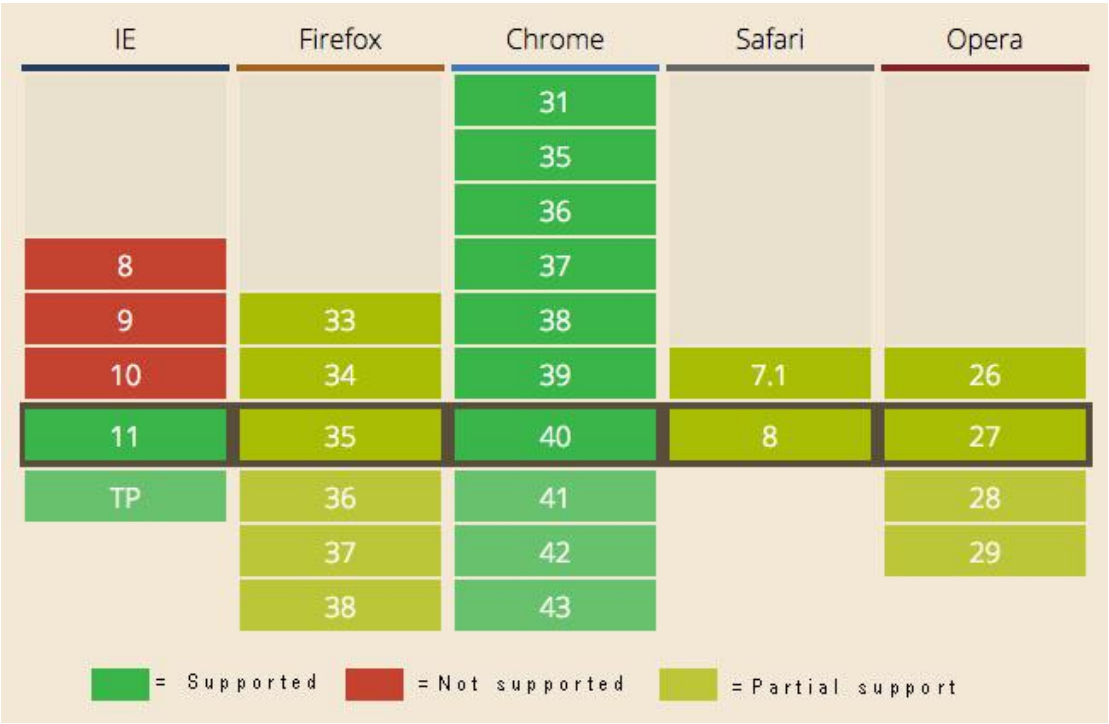
2.2.1 Βαθμίδα Παρουσίασης: Ο Φυλλομετρητής

Ο φυλλομετρητής είναι ένα λογισμικό που προβάλλει στον χρήστη δεδομένα από ιστοσελίδες ιστοτόπων στον Παγκόσμιο Ιστό, και του επιτρέπει την αλληλεπίδραση με αυτά. Για κάθε φυλλομετρητή διατίθενται αρκετά πρόσθετα στοιχεία(plug-ins) με στόχο την επαύξηση των δυνατοτήτων τους. Ένα βασικό θέμα για τις αποφάσεις που πάρθηκαν σε αυτή την διπλωματική ήταν ο στόχος να μην απαιτείται plug-in για την εφαρμογή μας. Οι φυλλομετρητές χρησιμοποιούν τη γλώσσα μορφοποίησης HTML για την προβολή των ιστοσελίδων, επομένως η εμφάνιση της διαφέρει ανάλογα με τον φυλλομετρητή. Στην αγορά, υπάρχει μεγάλος αριθμός φυλλομετρητών, εκ των οποίων οι πιο διαδεδομένοι είναι οι Chrome, Internet Explorer, Firefox, Safari, Opera (εικόνα 2.1).

Η επιλογή του φυλλομετρητή φυσικά δεν εξαρτάται από εμάς, αλλά από τον τελικό χρήστη που θα χρησιμοποιήσει την εφαρμογή μας. Δυστυχώς ανάμεσα στους φυλλομετρητές υπάρχουν ασυμφωνίες στον τρόπο με τον οποίο μεταφράζουν την CSS, η οποία είναι ουσιαστικά μία γλώσσα μορφοποίησης του κώδικα HTML. Για τον λόγο αυτό οι προγραμματιστές διαδικτυακών εφαρμογών προσπαθούν να καλύψουν όσο δυνατόν μεγαλύτερο αριθμό φυλλομετρητών. Στα πλαίσια αυτής της διπλωματικής προσπαθήσαμε να καλύψουμε τους πιο διαδεδομένους φυλλομετρητές καθώς είναι μία εφαρμογή που θα φιλοξενηθεί στον Παγκόσμιο Ιστό. Στα επόμενα υποκεφάλαια, θα εξηγήσουμε την επιλογή μας να χρησιμοποιήσουμε WebGL (Web Graphics Library). Η επιλογή μας αυτή μας περιόριζε αρχικά καθώς ο Internet Explorer δεν υποστήριζε την τεχνολογία WebGL, αλλά στην πορεία με την έκδοση Internet Explorer 11 που η σταθερή της έκδοση δόθηκε στο κοινό τον Δεκέμβριο του 2014 υποστηρίζεται (εικόνα 2.2).



Εικόνα 2.1- Δημοτικότητα των φυλλομετρητών Δεκέμβριος 2014 από το www.w3schools.com



Εικόνα 2.2- Οι εκδόσεις των φυλλομετρητών που υποστηρίζουν WebGL.

2.2.2 Βαθμίδα Εκτέλεσης: Ο Διακομιστής Ιστού (Web Server)

Ο χρήστης της διαδικτυακής εφαρμογής μέσω του φυλλομετρητή, στέλνει αιτήματα στον διακομιστή (ή αλλιώς εξυπηρετητή) ο οποίος υπολογίζει τα αποτελέσματα των ερωτημάτων αυτών και επιστρέφει την απάντηση στον χρήστη. Επομένως απαραίτητο δομικό στοιχείο μιας διαδικτυακής εφαρμογής είναι ο εξυπηρετητής που περιέχει το λογισμικό που απαιτείται από την εφαρμογή για να κάνει τους υπολογισμούς των αιτημάτων που δέχεται και να στείλει τις απαντήσεις

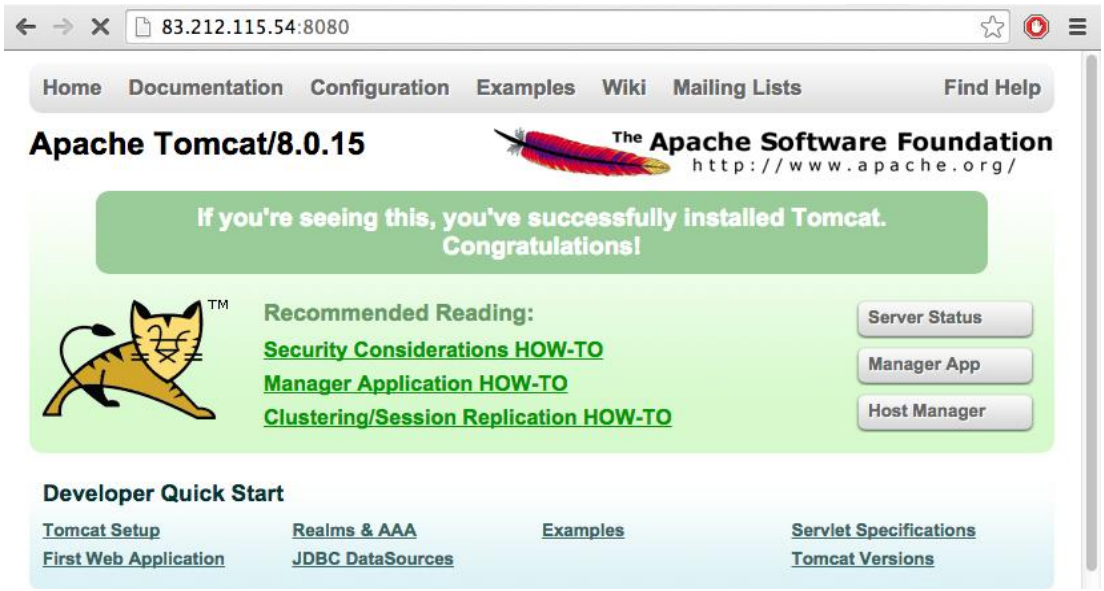
πίσω στον χρήστη. Δύο είναι οι βασικοί ανταγωνιστές στο μερίδιο της αγοράς στον τομέα των web-servers, ο Apache Server και ο IIS (Internet Information Services) της Microsoft.

Ο IIS αποτελεί την λύση της Microsoft στον τομέα των web-servers. Υποστηρίζει αρκετές διαδικτυακές τεχνολογίες και ιδιαίτερα εκείνες που αποτελούν δημιουργίες της Microsoft. Κατέχει το 33.04% της αγοράς (Απρίλιος 2014), και στα θετικά βρίσκουμε ότι συνεργάζεται άψογα με το λειτουργικό των Windows, σε αντίθεση με άλλους servers. Στα μειονεκτήματα του συγκαταλέγονται το γεγονός ότι δεν είναι δωρεάν, ότι δεν είναι ανοιχτού κώδικα και ότι τρέχει μόνο σε περιβάλλον Windows.(εικόνα 2.3)

Εικόνα 2.3- Αρχική σελίδα του διακομιστή IIS της Microsoft.

Ο Apache HTTP Server αποτελεί τον κυρίαρχο στο μερίδιο της αγοράς με 37.74%. Διατίθεται δωρεάν, ως project ανοιχτού κώδικα από το Apache Software Foundation και υποστηρίζει λειτουργικά συστήματα Windows και UNIX. Διακρίνεται για την σταθερότητα , την ταχύτητα καθώς και την ασφάλεια του. Επιπλέον ως ανοιχτού κώδικα, μπορεί να επεκταθεί και να γίνει ευέλικτος ανάλογα με την χρήση. Ο Apache Tomcat , έχει ενσωματωμένο τον Apache HTTP server και χρησιμοποιείται για την ανάπτυξη Java Servlets. Τα servlets είναι μια τεχνολογία που χρησιμοποιείται για την λήψη και την απάντηση σε αιτήματα (requests) από τον web client, συνήθως μέσω του πρωτοκόλλου HTTP. Πρόκειται για μικρά κομμάτια κώδικα που εκτελούνται στην πλευρά του web-server και απαραίτητα για την εφαρμογή που αναπτύσσουμε.

Επιλέξαμε επομένως να χρησιμοποιήσουμε τον Apache Tomcat διότι διακρίνεται για την σταθερότητα του και ταυτόχρονα είναι και δωρεάν. Επιπλέον η ευελιξία που παρέχει με την υποστήριξη Unix και Windows έπαιξε σημαντικό ρόλο καθώς δεν γνωρίζαμε εξ αρχής σε τι λειτουργικό σύστημα θα στηθεί η εφαρμογή. (εικόνα 2.4)



Εικόνα 2.4- Η σελίδα επιβεβαίωσης της σωστής εγκατάστασης του apache tomcat

2.2.3 Βαθμίδα Αποθήκευσης: Βάση δεδομένων

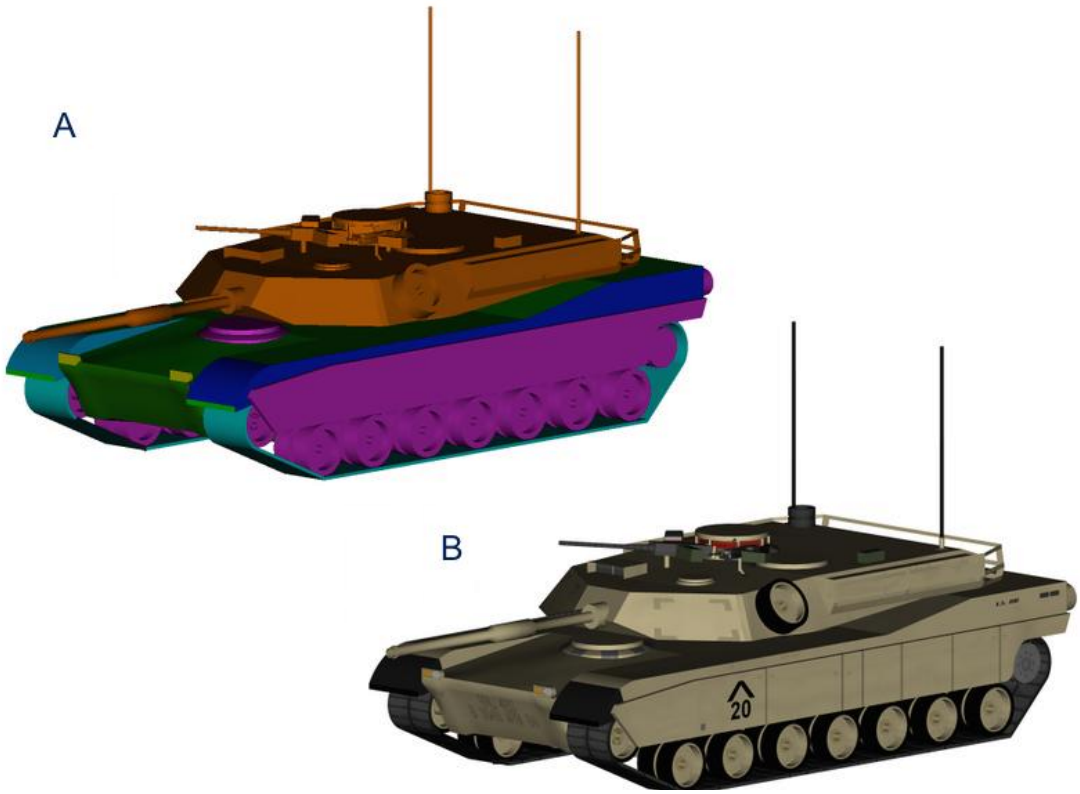
Η βάση δεδομένων είναι απαραίτητη όταν η εφαρμογή μας απαιτεί την αποθήκευση δεδομένων που δημιουργούνται από τον χρήστη. Τα συστήματα βάσεων δεδομένων έρχονται με την μορφή εξυπηρετητών (Database Servers) οι οποίοι συνεργάζονται με τον διακομιστή ιστού (web server). Συνήθως τα συστήματα αυτά υποστηρίζουν ερωτήματα (queries) προς την βάση δεδομένων. Υπάρχει πληθώρα συστημάτων βάσεων δεδομένων με πιο διαδεδομένες τις MySQL, PostgreSQL, Microsoft SQL Server, SAP και IBM DB2.

Στην διπλωματική αυτή χρησιμοποιήσαμε την MySQL, καθώς από τις πιο διαδεδομένες βάσεις δεδομένων, η οποία διατίθεται δωρεάν και διακρίνεται για την σταθερότητα της και τις δυνατότητες της. Χρησιμοποιείται σε μεγάλης κλίμακας διαδικτυακές εφαρμογές όπως το Facebook. Διατίθεται για UNIX και Windows και επιπλέον υπάρχουν βιβλιοθήκες για επικοινωνία με τον MySqlServer.

2.3 Τρισδιάστατα Μοντέλα

Τα τρισδιάστατα μοντέλα αντιπροσωπεύουν ένα πραγματικό αντικείμενο στον χώρο, χρησιμοποιώντας μία συλλογή από σημεία σε τρισδιάστατο χώρο τα οποία συνδέονται με διάφορες γεωμετρικές οντότητες όπως τρίγωνα και καμπύλες επιφάνειες. Όντας τα τρισδιάστατα μοντέλα μία συλλογή από δεδομένα (σημεία στον χώρο και άλλες πληροφορίες) μπορούν να δημιουργηθούν είτε με προγράμματα σχεδίασης τρισδιάστατων αντικειμένων και σκηνών, είτε με την χρήση τρισδιάστατου σαρωτή (3D scanner).

Ένα τρισδιάστατο μοντέλο περιγράφεται πιο συγκεκριμένα από τα σημεία στον τρισδιάστατο χώρο που ονομάζονται vertices (κορυφές), τα οποία συνδέονται με γραμμές και έτσι δημιουργούνται τα πολύγωνα. Κάθε πολύγωνο μπορεί να έχει ένα υλικό που χαρακτηρίζεται από χρώμα, διαφάνεια και άλλες παραμέτρους, ενώ το υλικό μπορεί να χρησιμοποιεί μία εικόνα(texture) για πιο αληθοφανές αποτέλεσμα(εικόνα 2.5).



Εικόνα 2.5- Α: μοντέλο χωρίς χρήση εικόνας στο υλικό. Β: Μοντέλο με χρήση εικόνας στο υλικό

Οι κάρτες γραφικών πλέον διαθέτουν τεράστια υπολογιστική δύναμη, και τα προηγμένα λογισμικά σχεδίασης τρισδιάστατων σκηνών (3DS MAX, Blender,κ.α) με αλγορίθμους φωτισμού παρέχουν πλέον απεριόριστες δυνατότητες στους σχεδιαστές- γραφίστες να δημιουργήσουν φωτορεαλιστικό αποτέλεσμα που μπορεί να μπερδέψει το ανθρώπινο μάτι, και να πειστεί ότι είναι πραγματική εικόνα (εικόνα 2.6).

Υπάρχουν πάρα πολλοί τύποι αρχείων τρισδιάστατων σκηνών, ενδεικτικά αναφέρουμε τους εξής : collada(.dae), .x3d, .dwf, .3ds, wavefront(.obj). Υπάρχουν πολλές διαφορές μεταξύ των αρχείων αυτών σε σχέση με την μορφοποίηση που έχει το αρχείο (XML, ASCII, Binary, κα) και με το μέγεθος της πληροφορίας που καταγράφει ο κάθε τύπος αρχείου.

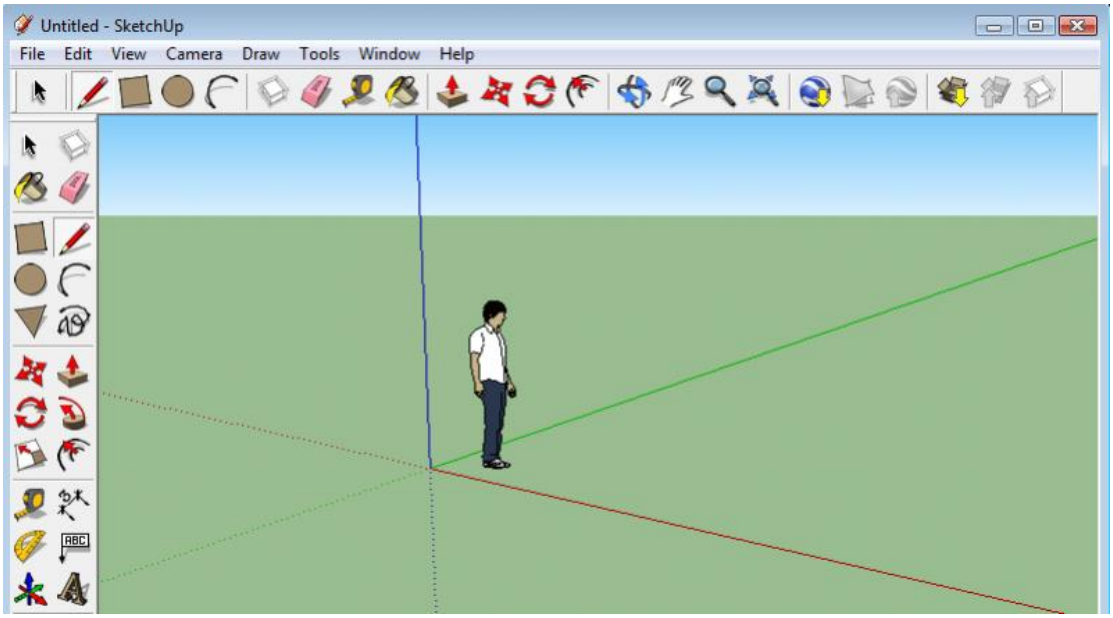
Ο τύπος του αρχείου που επιλέχθηκε έχει σημασία καθώς θα πρέπει να μπορεί να γίνει εξαγωγή του μοντέλου από το πρόγραμμα σχεδιασμού που επιλέχτηκε (θα αναφερθούμε στο επόμενο υποκεφάλαιο), καθώς επίσης και να υποστηρίζεται από την εφαρμογή μας, δηλαδή να μπορεί να γίνει πιστή αποτύπωση του μοντέλου στον φυλλομετρητή σε σχέση με το μοντέλο που σχεδιάστηκε.

Ενώ αρχικά είχαμε επιλέξει να εξάγουμε τα μοντέλα σε .dae, αναθεωρήσαμε καθώς το πλεονέκτημα τους, δηλαδή η μεγάλη λεπτομέρεια που εμπεριείχαν, φάνηκε εμπόδιο, καθώς ήταν πολύ μεγάλα σε όγκο, πράγμα απαγορευτικό για διαδικτυακή εφαρμογή, αλλά επίσεις είναι πολύ πολύπλοκα δομημένα με αποτέλεσμα να χρειάζονται αρκετή ώρα για να φορτωθούν και να απεικονιστούν στην εφαρμογή. Έτσι καταλήξαμε στα .obj, καθώς είναι αρκετά διαδεδομένα, με αποτέλεσμα να έχουν αναπτυχθεί πολλά θέματα συζητήσεων γύρω από αυτά στο διαδίκτυο, και καλύπταν τις απαιτήσεις μας, δηλαδή να είναι μικρά σε όγκο , αλλά και να μην είναι πολύπλοκη η δομή τους.

Εικόνα 2.6- Αριστερά βλέπουμε εικόνα από τρισδιάστατη σκηνή κατασκευασμένη στο 3DS MAX. Δεξιά το πλέγμα των πολυγώνων (wireframe) που χρειάστηκαν για την αναπαράστασή της.

2.4 Google SketchUp

Στην διπλωματική που υλοποιήσαμε, είχαμε το πλεονέκτημα ότι τα τρισδιάστατα μοντέλα που αναπαριστούν τα μνημεία, τα δημιούργησαν οι φοιτητές της αρχιτεκτονικής υπό την επιτήρηση των καθηγητών τους. Η πρόταση από την μεριά τους ήταν να χρησιμοποιηθεί το Google SketchUp ως εργαλείο για την σχεδίαση των τρισδιάστατων μοντέλων (εικόνα 2.7).



Εικόνα 2.7- Η αρχική σκηνή του Google SketchUp

Το Google SketchUp παρέχει την δυνατότητα να εξάγει τα μοντέλα σε αρχεία τύπου: .skp, .obj, και .dae. Όπως αναφέραμε και παραπάνω επιλέχτηκαν τα .obj, τα οποία συνοδεύονται από άλλο ένα αρχείο, το .mtl το οποίο περιγράφει σε ποιές επιφάνειες πρέπει να χρησιμοποιηθεί κάθε υλικό που περιέχει.

Για να είναι υλοποιήσιμη η εφαρμογή στον πραγματικό κόσμο, με τις υπάρχουσες ταχύτητες του Ίντερνετ και τις υπάρχουσες κάρτες γραφικών έπρεπε να ορίσουμε

ορισμένες προδιαγραφές. Τέτοιες προδιαγραφές ήταν το μέγεθος του αρχείου, στοιχείο πολύ σημαντικό για διαδικτυακή εφαρμογή, καθώς ο χρήστης του διαδικτύου δεν θα περιμένει πάνω από ένα με δύο δευτερόλεπτα για να φορτωθεί η εφαρμογή του. Θέσαμε ώς ανώφλι το 1Mb, και για την πολυπλοκότητα του μοντέλου τα εκατό χιλιάδες πολύγωνα.

2.5 Τεχνολογίες απεικόνισης τρισδιάστατων γραφικών σε περιβάλλον φυλλομετρητή

Το πιο σημαντικό μέρος της έρευνας που έγινε για την υλοποίηση της διπλωματικής ήταν η επιλογή της τεχνολογίας που θα κάλυπτε τις απαιτήσεις μας για την απεικόνιση τρισδιάστατων γραφικών στον φυλλομετρητή. Παρακάτω παρουσιάζονται τεχνολογίες που επιτρέπουν την ανάπτυξη τέτοιων εφαρμογών.

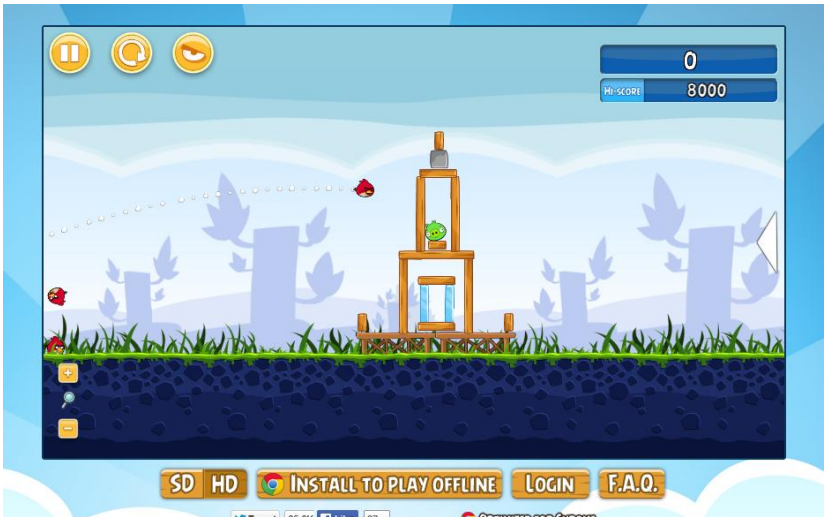
2.5.1 Java Applet με χρήση βιβλιοθήκης Java 3D

Για την δημιουργία τρισδιάστατης σκηνής στον φυλλομετρητή θα μπορούσε να χρησιμοποιηθεί η βιβλιοθήκη java3d μέσα στα πλαίσια ενός επιπρόσθετου προγράμματος που πρέπει να εγκατασταθεί στον υπολογιστή του χρήστη(applet). Η συγκεκριμένη συγκεκριμένη βιβλιοθήκη επιτρέπει την δημιουργία πολύπλοκων τρισδιάστατων σκηνών οι οποίες είναι επιταχυμένες από την κάρτα γραφικών. Τρέχει είτε πάνω στην OpenGL είτε την Direct3D. Στα πλεονεκτήματα συγκαταλέγεται η Java ως γλώσσα προγραμματισμού, από τις πιο διαδεδομένες γλώσσες, άλλα βασικό της μειονέκτημα είναι η χρήση applet, καθώς δεν καλύπτει την προϋπόθεση της διπλωματικής.

Εικόνα 2.8- Runescape, ένα διαδικτυακό τρισδιάστατο παιχνίδι υλοποιημένο με java3d

2.5.2 Flash player

Ο Flash player μπορεί να τρέξει στον φυλλομετρητή με την χρήση επιπρόσθετου προγράμματος (plug-in) σε Unix και σε Windows. Μέχρι την έκδοση 10 το flash δεν είχε την δυνατότητα της χρήσης της κάρτας γραφικών. Με την έκδοση 11 ο flash player χρησιμοποιούσε την επιτάχυνση της κάρτας γραφικών, σε ίδιο βαθμό με την WebGL. Η τωρινή έκδοση είναι η 16, και η εταιρία που την παρέχει (η Adobe) ισχυρίζεται ότι πάνω από τετρακόσια εκατομμύρια χρήστες την κατέβασαν στις πρώτες έξι εβδομάδες έκδοσης της. Είναι αρκετά διαδεδομένος και χρησιμοποιείται κατά κόρον σε παιχνίδια στο facebook. Υπάρχουν διάφορες μηχανές γραφικών οι οποίες εκτελούνται χρησιμοποιώντας flash player, αλλά καθώς απαιτείται plugin δεν προχωρήσαμε σε παιρεταίρω διερεύνηση.(εικόνα 2.9)



Εικόνα 2.9- AngryBirds, από τα πιο διαδεδομένα παιχνίδια, βασισμένο στον Flash Player

2.5.3 Unity 3D

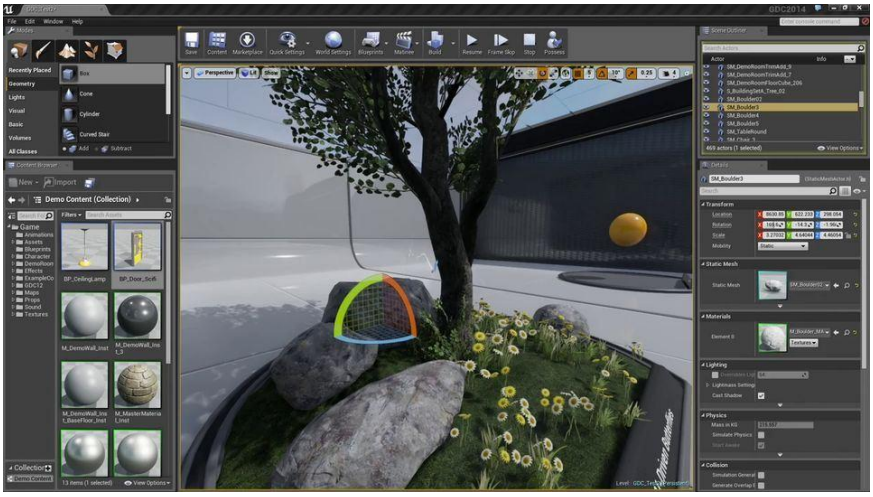
Η Unity 3D αποτελεί ένα ολοκληρωμένο περιβάλλον ανάπτυξης παιχνιδιών που παρέχει πληθώρα εργαλείων για γρηγορότερη ανάπτυξη και αποσφαλμάτωση. Ορισμένα από αυτά είναι η γραφική διεπαφή για τον έλεγχο των αντικειμένων, πρόσβαση σε όλες τις λεπτομέρειες και ιδιότητες της γεωμετρίας και άμεση προεπισκόπηση του παιχνιδιού. Υποστηρίζει τις γλώσσες C# και Javascript, το οποίο συγκαταλέγεται στα πλεονεκτήματα μαζί με την πολύ ενεργή κοινότητα χρηστών που διαθέτει. Υπάρχει τέλος η επιλογή μετατροπής της εργασίας για φυλλομετρητή, αλλά για να υποστηριχτεί από τον φυλλομετρητή απαιτείται η χρήση επιπρόσθετου προγράμματος. Κυκλοφορεί δοκιμαστικά η έκδοση 5 της Unity, είναι άγνωστη η ημερομηνία σταθερής έκδοσης, η οποία υποστηρίζει την δημιουργία παιχνιδιών σε browser χωρίς την χρήση plugin, μέσω της WebGL. (εικόνα 2.10)

Εικόνα 2.10- Το περιβάλλον της Unity, κατά την δημιουργία τρισδιάστατης σκηνής παιχνιδιού.

2.5.4 Unreal Engine

Η Unreal Engine είναι και αυτή όπως η Unity μία μηχανή παιχνιδιών με ολοκληρωμένο περιβάλλον ανάπτυξης. Τα παιχνίδια που παράγονται μπορούν να

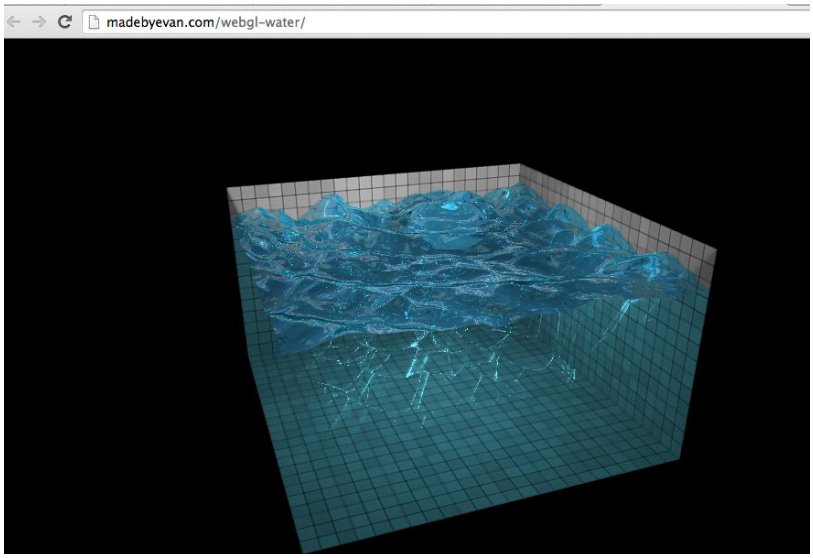
τρέξουν σε πολλά λειτουργικά συστήματα και κονσόλες παιχνιδιών. Διαθέτει δική της γλώσσα προγραμματισμού, την Unreal Script. Με την τελευταία της έκδοση στα μέσα του 2014 παρέχει σε συνεργασία με τον Firefox την δημιουργία τρισδιάστατων σκηνών για το στοιχείο canvas της HTML5. Δυστυχώς η έκδοση αυτή βγήκε κατά την περάτωση της υλοποίησης της εφαρμογής με αποτέλεσμα να μην μπορέσουμε να την αξιοποιήσουμε.



Εικόνα 2.11- Το περιβάλλον της Unreal Engine.

2.5.5 WebGL

Η WebGL είναι ουσιαστικά μία διεπαφή λογισμικού(API-Application Programming Interface) της JavaScript για την πρόσβαση στην κάρτα γραφικών μέσω φυλλομετρητή. Παρέχει διαδραστική απεικόνιση τρισδιάστατων και δισδιάστατων γραφικών με επιτάχυνση υλικού σε συμβατούς φυλλομετρητές χωρίς την χρήση επιπρόσθετων προγραμμάτων. Συμβατοί είναι όλοι οι βασικοί φυλλομετρητές, όπως ο Chrome, ο Firefox, ο Safari, ο Opera και ο Internet Explorer 11. Η WebGL βασίζεται στην OpenGL ES 2.0 η οποία αποτελεί ένα υποσύνολο της OpenGL(Open Graphics Library)και παρέχει πρόσβαση στην κάρτα γραφικών για τις αυτόνομες εφαρμογές του υπολογιστή. Η WebGL περιλαμβάνει ρουτίνες σε javascript για τον προγραμματισμό των τρισδιάστατων γραφικών, και χρησιμοποιεί το στοιχείο canvas της HTML5 για την απεικόνισή τους στον φυλλομετρητή. Στα θετικά συγκαταλέγεται ότι δεν χρειάζεται επιπρόσθετο λογισμικό, και ότι καλύπτεται από όλους τους βασικούς φυλλομετρητές. Στα αρνητικά είναι ότι η WebGL είναι χαμηλού επιπέδου προγραμματισμού, επομένως και αρκετά δυσνόητη. Για τον λόγω αυτό υπάρχουν πολλές βιβλιοθήκες με ρουτίνες Javascript που κάνουν την χρήση της αρκετά πιο εύκολη. Μερικές από αυτές είναι οι Three.js, SceneJS, Babylon.js.



Εικόνα 2.12- Τρισδιάστατη διαδραστική σκηνή σε φυλλομετρητή με χρήση της WebGL

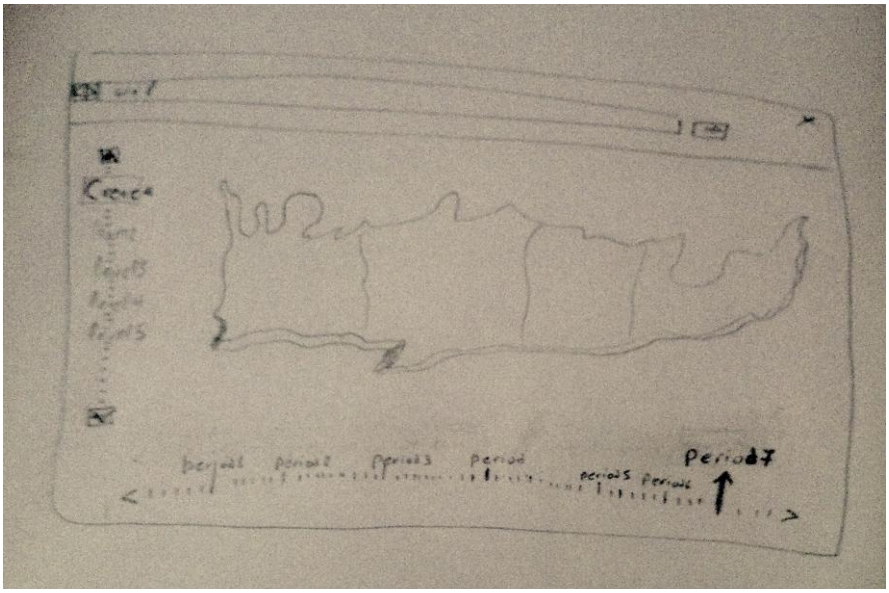
2.5.6 Αιτιολόγηση επιλογής της WebGl

Επιλέξαμε να χρησιμοποιήσουμε την WebGL, καθώς καλύπτει την βασική μας προϋπόθεση, να μην απαιτείται επιπρόσθετο λογισμικό στον φυλλομετρητή για να λειτουργεί η εφαρμογή μας. Μπορεί το αποτέλεσμα να υστερεί σε σχέση με τα αποτελέσματα που βγαίνουν κυρίως από τις μηχανές παιχνιδιών, Unity και Unreal, αλλά είναι πολύ σημαντικό να τονίσουμε ότι τρέχει στον φυλλομετρητή χωρίς τίποτα επιπρόσθετο, αρκεί να είναι συμβατή η κάρτα γραφικών. Λόγω του χαμηλού επιπέδου προγραμματισμού που απαιτείται για την WebGL, χρησιμοποιήσαμε βιβλιοθήκες που μας παρείχαν σημαντικές συναρτήσεις για την υλοποίηση της εφαρμογής.

Επιλέξαμε την Three.js ,καθώς ήταν η πιο διαδεδομένη, και λόγω της καλύτερης τεκμηρίωσης των συναρτήσεων της. Καθώς ακόμη είναι σε αρχικά στάδια, έπαιξε πολύ σημαντικό ρόλο η κοινότητα χρηστών που είχε κάθε βιβλιοθήκη, και πόσο ενεργοποιημένη ήταν.

2.6 Γραφική Διεπαφή Χρήστη

Η έρευνα μας γύρω από εξ ολοκλήρου τρισδιάστατες εφαρμογές στο διαδίκτυο μας έδειξε ότι βασικό χαρακτηριστικό ήταν να καταλαμβάνει όσο μεγαλύτερο χώρο της οθόνης γίνεται το στοιχείο canvas, που πάνω του αναπτύσσεται η τρισδιάστατη σκηνή. Πολύ σημαντικό ρόλο, έπαιξαν και οι οδηγίες από τον κύριο Παρθένιο ο οποίος μας έδωσε μία εικόνα όπως φανταζόταν την εφαρμογή, και εμείς την μετατρέψαμε σε κώδικα. Οδηγηθήκαμε λοιπόν στο να είναι όλη η οθόνη η τρισδιάστατη σκηνή, και να υπάρχουν στην ουσία μόνο δύο μπάρες πλοήγησης, μία στα αριστερά που μας δείχνει σε ποιό επίπεδο λεπτομέρειας βρισκόμαστε, και μία στο κάτω μέρος της οθόνης, η οποία μας δείχνει σε ποιά χρονική περίοδο βρισκόμαστε(εικόνα 2.13). Έχοντας σαν παραδείγματα μόνο κάποια demo που υπάρχουν από διαδικτυακές εφαρμογές με τρισδιάστατα γραφικά, δεν μπορούσαμε να έχουμε κάποιο μέτρο σύγκρισης. Το βασικό και μόνο παράδειγμα που είχαμε από πλήρη τρισδιάστατη πλοήγηση στο διαδίκτυο, είναι το Google StreetView, το οποίο σε όλο το παράθυρο του φυλλομετρητή έχει την τρισδιάστατη σκηνή του.



Εικόνα 2.13- Πρώτο πρωτότυπο χαρτιού (Paper-prototype)

3 Τεχνολογική Βάση

3.1 Εισαγωγή

Στο προηγούμενο κεφάλαιο έγινε αναφορά στην σχετική έρευνα που πραγματοποιήθηκε για τις τεχνολογίες οι οποίες θα μας βοηθούσαν στην υλοποίηση συγκεκριμένων τμημάτων της εφαρμογής. Αφού παρουσιάστηκε ένας αριθμός διαφορετικών τεχνολογιών και μεθοδολογιών ανάπτυξης, με τα πλεονεκτήματα και μειονεκτήματα τους, υιοθετήθηκαν και χρησιμοποιήθηκαν συγκεκριμένες από αυτές. Στο κεφάλαιο αυτό θα γίνει αναλυτική περιγραφή της αρχιτεκτονικής και της λειτουργίας των τεχνολογιών, των εργαλείων και των μεθοδολογιών που χρησιμοποιήθηκαν για την ανάπτυξη της εφαρμογής.

3.2 Πλευρά Χρήστη-Πελάτη (Client side)

Στην πλευρά του χρήστη-πελάτη της εφαρμογής, ουσιαστικά εννοείται ο φυλλομετρητής που χρησιμοποιεί ο κάθε χρήστης, αποστέλονται μέσω του διαδικτύου από τον διακομιστή κώδικας HTML,Javascript, και CSS. Ο φυλλομετρητής μπορεί και «διαβάζει» αυτόν τον κώδικα, ώστε ο χρήστης να δεί το περιεχόμενο που πρέπει. Στην συνέχεια θα εξηγήσουμε τί κάνει ο κάθε κώδικας και για ποιον λόγο χρησιμοποιείται.

3.2.1 HTML5

Η HTML5 αποτελεί μία γλώσσα σήμανσης (MarkupLanguage) για την δόμηση και την παρουσίαση του περιεχομένου ιστοσελίδας για τον Παγκόσμιο Ιστό. Από το 2014 η HTML5 είναι η τελευταία έκδοση της HTML, με την προηγούμενη έκδοση να είχε δημιουργηθεί το 1997. Βασικός σκοπός της HTML5 είναι η μείωση της ανάγκης για επιπρόσθετα προγράμματα στον φυλλομετρητή για την αναπαραγωγή πολυμέσων.

Η HTML παρέχει μέσα για την δημιουργία ορθά δομημένων εγγράφων υποδηλώνοντας σημασιολογικά στοιχεία κειμένου όπως παράγραφοι, επικεφαλίδες, λίστες, πίνακες, σύνδεσμοι και πολλά άλλα. Τα στοιχεία της HTML λέγονται HTML elements.Στην HTML5 ειδικότερα προστέθηκαν νέα στοιχεία όπως το <video>, <audio> και το <canvas> element. Τα στοιχεία αυτά συμπεριλήφθηκαν ώστε να δοθεί η δυνατότητα δημιουργίας πλούσιου περιεχομένου Ιστοσελίδας χωρίς την χρήση ιδιόκτητων επιπρόσθετων προγραμμάτων στον φυλλομετρητή. Εκμεταλλευθήκαμε λοιπόν για αυτή την διπλωματική το στοιχείο <canvas> της HTML5 το οποίο η WebGL εκμεταλλεύτηκε για να αποκτήσει πρόσβαση στην κάρτα γραφικών του υπολογιστή και να απεικόνιση τα τρισδιάστατα γραφικά.

Η δήλωση ενός html element αποτελείται από τα εξής βασικά συστατικά:

- Ένα ζεύγος tagstου στοιχείου: το tagανοίγματος και το tagκλεισίματος.
- Ένα σύνολο από attributesτα οποία δηλώνονται μέσα στα tagstου στοιχείου
- Το περιεχόμενο του στοιχείου προς παρουσίαση στην οθόνη του φυλλομετρητή το οποίο μπορεί να έχει είτε τη μορφή κειμένου, είτε γραφικών.

Ένα HTMLστοιχείο είναι οτιδήποτε περιλαμβάνεται μεταξύ των δύο tags. Το tagτου στοιχείου είναι στην ουσία το είδος του στοιχείου εσώκλειστο μεταξύ δύο αγκυλών. Το tagκλεισίματος είναι όπως το tagανοίγματος, αλλά με μία κάθετο

αμέσως πριν το όνομα toutag.Υπάρχουν πάρα πολλές επιλογές μεταξύ των στοιχείων της HTML, μερικές από αυτές αναφέρονται στον παρακάτω πίνακα. (πίνακας 3.1)

Tag	Περιγραφή
<a>	Ορίζει έναν υπερσύνδεσμο (hyperlink)
** **	Ορίζει ένα τμήμα μέσα στην ιστοσελίδα
<h1> to <h6>	Ορίζει επικεφαλίδες (6 είδη)
** **	Ορίζει μία παράγραφο
<select>	Ορίζει μια drop-down λίστα
<table>	Ορίζει έναν πίνακα
<tr>	Ορίζει μία γραμμή στον πίνακα
<td>	Ορίζει ένα κελί μέσα στον πίνακα
<img>	Ορίζει μία εικόνα
<canvas>	Χρησιμοποιείται για την αποτύπωση γραφικών μέσω της javascript.

Πίνακας 3.1- Ενδεικτική περιγραφή των tagsτης html.

Θα δώσουμε ορισμένα παραδείγματα χρήσης των tags. Αν θα θέλαμε να θέσουμε σαν περιεχόμενο της ιστοσελίδας μία παράγραφο με επικεφαλίδα, τότε θα πρέπει να εσωκλείσουμε το περιεχόμενο της επικεφαλίδας με το tag : <h1></h1>, και το περιεχόμενο της παραγράφου με το tag : όπως στο παρακάτω παράδειγμα (εικόνα 3.1)

Εικόνα 3.1- Παράδειγμα χρήσης των tagselementsτης HTML.

Μέσα στα tagsανοίγματος μπορούν να υπάρχουν και attributesπου περιγράφουν παραμέτρους που επηρεάζουν το στοιχείο που περιγράφεται. Για παράδειγμα αν θέλουμε να εμφανίζεται μία εικόνα στον φυλλομετρητή με διαστάσεις 50x50 pixels τότε πρέπει να χρησιμοποιήσουμε το tag<img>και να ορίσουμε τα attributessrc(το path στο οποίο βρίσκεται η εικόνα), height (το ύψος της εικόνας), και width(το πλάτος της εικόνας). (εικόνα 3.2)

Html Code:

```
<img src="smiley.gif"  width="50" height="50">
```

Result:

Εικόνα 3.2- Παράδειγμα χρήσης του imgtagτης html.

Ένα html element μπορεί να περιλαμβάνει μέσα στο περιεχόμενο του ένα άλλο html element. Στο παρακάτω παράδειγμα (εικόνα 3.3) βλέπουμε ότι για να δημιουργήσουμε ένα πίνακα πρέπει να ορίσουμε τον πίνακα, και μέσα σε αυτόν βρίσκεται εμφωλευμένο το όρισμα γραμμής πίνακα μέσα στην οποία βρίσκεται εμφωλευμένο το όρισμα κελιών του πίνακα. Στο συγκεκριμένο παράδειγμα, δώσαμε το attribute “ border=’1’ ” ώστε να φαίνονται τα όρια των κελιών, ειδάλλως ως αποτέλεσμα θα είχαμε στοιχισμένο μόνο το περιεχόμενο, που δεν παραπέμπει στον πίνακα που έχουμε συνηθίσει να εννοούμε.

Html Code:

```
<table border="1">
  <tr>
      <td>1st row/1st cell</td>
      <td>1st row/2nd cell</td>
  <tr>
  <tr>
      <td>2nd row/1st cell</td>
      <td>2nd row/2nd cell</td>
  <tr>
<table>
```

Result:

1st row/1st cell	1st row/2nd cell
2nd row/1st cell	2nd row/2nd cell

Εικόνα 3.3- Παράδειγμα χρήσης του <table>tag με εμφωλευμένα τα <tr>και <td>.

Κάθε έγγραφο της ιστοσελίδας σε γλώσσα htmlαπαιτεί την ύπαρξη κάποιων συγκεκριμένων tagsπου καθορίζουν την βασική δομή του. Καταρχήν υπάρχει το <html>element, το οποίο εσωκλείει μέσα του όλο το περιεχόμενο της σελίδας. Το html element χρησιμοποιείται για να αναγνωρίζει ο φυλλομετρητής ότι το περιεχόμενο που εσωκλείεται αποτελεί περιγραφή ιστοσελίδας σε γλώσσα HTML. Το html element αποτελεί τη ρίζα (root element) στην δεντρική δομή του αρχείου. Μέσα στο html element υπάρχουν εμφωλευμένα αρκετά στοιχεία όπως το head element, το οποίο αποτελεί «επικεφαλίδα» του εγγράφου. Στο head element περιλαμβάνεται εμφωλευμένο το title element το οποίο περιέχει τον τίτλο της ιστοσελίδας, ο οποίος φαίνεται και στην καρτέλα του φυλλομετρητή. Επίσεις στο head elementυπάρχουν και αναφορές σε αρχεία που χρησιμοποιούνται από το αρχείο αυτό όπως τα αρχεία Javascriptκαι CSS, στα οποία θα αναφερθούμε στα επόμενα κεφάλαια. Ένα άλλο βασικά στοιχείο του html element είναι το body element, το οποίο αποτελεί το δοχείο για όλα εκείνα τα στοιχεία του περιεχομένου της σελίδας που είναι εμφανίσιμα στην οθόνη του φυλλομετρητή.

Ακόμη, για κάθε elementτης htmlγλώσσας υπάρχουν κάποια βασικά attributes (γνωρίσματα) τα οποία περιγράφονται παρακάτω:

- Id: Toidπροσδίδει στο συγκεκριμένο στοιχείο ένα αναγνωριστικό όνομα το οποίο είναι μοναδικό σε όλο το έγγραφο html. Το αναγνωριστικό αυτό

όνομα, μπορεί να χρησιμοποιηθεί από τα css αρχεία για να γίνει αναφορά στο συγκεκριμένο htm lelement και να αλλαχτεί η εμφάνισή του, καθώς και από κώδικα Javascriptγια να υλοποιηθούν διεργασίες που αφορούν μόνο το συγκεκριμένο στοιχείο.

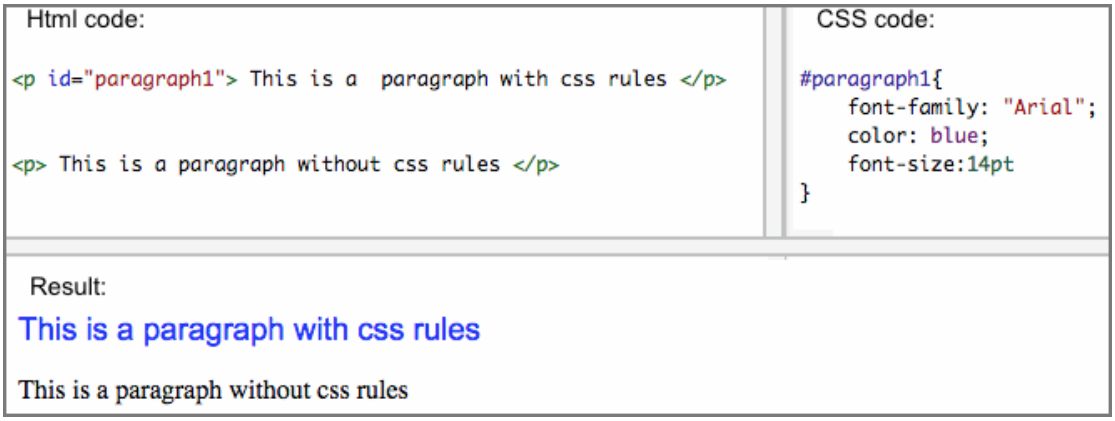
- Class: Με το συγκεκριμένο γνώρισμα μπορούμε να ομαδοποιήσουμε html elements που θέλουμε να συμπεριφέρονται συγκεκριμένα ή να εμφανίζονται με τον ίδιο τρόπο στον φυλλομετρητή. Για παράδειγμα μπορούμε να ορίσουμε σε ορισμένες παραγράφους την κλάση “attention”, και μέσω του cssνα προσδιορίσουμε ότι όσα elementφέρουν την κλάση αυτή θα έχουν Bold, κόκκινα γράμματα
- Title: Το γνώρισμα αυτό μπορεί να τοποθετηθεί σε κάποιο στοιχείο και να προσάψει μία γενική περιγραφή η οποία εμφανίζεται από τον browser ως tooltipόταν τοποθετείται ο κέρσορας πάνω από το συγκεκριμένο στοιχείο. Είναι αρκετά συνηθισμένο κυρίως στις εικόνες που χρησιμοποιούνται ως κουμπιά μετάβασης αλλά δεν είναι από όλους εμφανές τι προσδιορίζει η εκάστοτε εικόνα-κουμπί.

3.2.2 CSS

Η CSS(Cascading Style Sheets) είναι μία γλώσσα που χρησιμοποιείται για τον έλεγχο της εμφάνισης ενός εγγράφου που έχει γραφτεί με μία γλώσσα σήμανσης. Χρησιμοποιείται δηλαδή για τον έλεγχο της εμφάνισης ενός εγγράφου που γράφτηκε σε HTML και ουσιαστικά για την μορφοποίηση της εμφάνισης μίας ιστοσελίδας. Κάθε κανόνας CSS, στοχεύει ένα συγκεκριμένο στοιχείο με βάση το id του (γνώρισμα του στοιχείου), ή ένα σύνολο στοιχείων με βάση την κλάση τους ή με βάση το tagτους. Περιέχει κανόνες οι οποίοι αφορούν το χρώμα, την στοίχιση, την διευθέτηση ορίων, την μορφή του κέρσορα, και πολλά άλλα. Μπορεί ενταχθεί στον κώδικα της html ώς element ανάμεσα στα tags<style>…</style>, αλλά είναι προτιμότερο και «πιο σωστό» προγραμματιστικά να βρίσκεται σε ανεξάρτητο αρχείο, το οποίο καλείται από την HTML.

Παρακάτω δίνονται κάποια παραδείγματα για να εξεταστούν διάφορες μορφές που έχουν οι κανόνεςcss.

Έστω ότι θέλουμε να καθορίσουμε την γραμματοσειρά, το χρώμα και το μέγεθος κειμένου μίας παραγράφου με id=”paragraph1”. Ο κανόνας css θα είναι ο εξής:



Εικόνα 3.4- Παράδειγμα χρήσης κώδικα css.

Στο παραπάνω παράδειγμα βλέπουμε ότι για να επιλέξουμε να μορφοποιήσουμε συγκεκριμένο html elementμε κανόνες css , πρέπει να χρησιμοποιήσουμε το id του. Για να επιλέξουμε με βάση το id, προηγείται το σύμβολο “#”, ενώ αν θέλαμε να ορίσουμε κανόνες μορφοποίησης για μια ομάδα στοιχείων με βάση την κλάση τους, θα προηγούνταν του ονόματος της κλάσης μία τελεία (.class_name). Τέλος αν θα θέλαμε να θέσουμε κανόνα με βάση το tag, τότε δεν χρειάζεται κανένα σύμβολο μπροστά από το όνομα του tag.Υπάρχουν πάρα πολλές επιλογές για μορφοποίηση του htmlπεριεχομένου. Μπορεί να αφορούν την θέση του στοιχείου, το χρώμα των γραμμάτων, το φόντο του, τα όρια και τι είδους, καθώς και την σκίαση τους και πολλά άλλα.

3.2.3 Javascript

Η Javascript είναι η πιο δημοφιλής scripting γλώσσα για το διαδίκτυο. Χρησιμοποιείται σχεδόν σε όλες τις ιστοσελίδες προσθέτοντας λειτουργικότητα (επικύρωση στοιχείων, διάκριση browser, επικοινωνία με τον server, αντίδραση σε συμβάντα, δημιουργία cookies, ανάγνωση και εγγραφή HTML στοιχείων κ.α), και υποστηρίζεται απ’ όλους τους σύγχρονους browsers (Internet Explorer, Firefox, Chrome, Opera, Safari). Ο κώδικας της μπορεί να ενσωματωθεί σε ένα HTML αρχείο ή να κληθεί από ένα ή περισσότερα εξωτερικά αρχεία. Επίσης η χρήση της είναι ελεύθερη, δηλαδή δεν απαιτείται άδεια για την χρήση της στις εμπορικές εφαρμογές όπως και όλες οι τεχνολογίες που χρησιμοποιήθηκαν στην ανάπτυξη της εφαρμογή μας.

Αν και ξεκίνησε αρχικά σαν μια απλή scriptingγλώσσα για την προσθήκη μικρών δυναμικών λειτουργιών στις ιστοσελίδες, έχει εξελιχθεί σε μία ισχυρή γλώσσα προγραμματισμού με υποστήριξη δυναμικών συναρτήσεων, δομών οντοκεντρικού προγραμματισμού, μεθόδους που εκτελούνται ταυτόχρονα, ασύγχρονη επικοινωνία με τον server. Η Javascriptεκτελείται δυναμικά. Αυτό σημαίνει ότι ο κώδικας δεν μεταγλωττίζεται στο σύνολό του πριν την έναρξη της εκτέλεσης, αλλά διερμηνεύεται εντολή προς εντολή σε πραγματικό χρόνο. Αυτό επιτρέπει ιδιαίτερες τεχνικές δυναμικού προγραμματισμού, που δεν τις συναντάμε σε άλλες «παραδοσιακές» γλώσσες, αλλά ταυτόχρονα έχουμε επιβάρυνση στην ταχύτητα εκτέλεσης. Ωστόσο με την εξέλιξη των φυλλομετρητών, έχουν εμφανιστεί μηχανές εκτέλεσης της javascript οι οποίες επιτυγχάνουν όλο και μεγαλύτερες ταχύτητες εκτέλεσης. Αυτό έχει ως αποτέλεσμα τα προγράμματα javascriptστις ιστοσελίδες να γίνονται όλο και πιο περίπλοκα, με δυνατότητες ανάλογες προγραμμάτων desktopεφαρμογών.

Η σύνταξη της Javascriptμοιάζει με την σύνταξη της Cκαι της JAVA. Ο κώδικας δομείται μέσα σε blocks τα οποία ορίζονται από τις αγκύλες { } . Επιτρέπονται statementsόπως while, for, if,switch. Αν ο κώδικας έχει πάνω από μία εντολή javascript τότε πρέπει η κάθε εντολή να ακολουθείται από ελληνικό ερωτηματικό ¨;¨. Η Javascriptεπιτρέπει δυναμικούς τύπους μεταβλητών. Αυτό σημαίνει ότι μία μεταβλητή η οποία αρχικά έχει αριθμητική τιμή, μπορεί στην συνέχεια να της δοθεί μία αλφαριθμητική τιμή χωρίς να έχει γίνει κάποιο συντακτικό λάθος. Αυτό συμβαίνει διότι τα πάντα στην javascript αποτελούν αντικείμενα (objects). Τα αντικείμενα της javascript είναι στην ουσία πίνακες συσχετισμού. Παραδείγματος χάριν, έστω ότι ένα αντικείμενο που περιγράφει την οντότητα “πρόσωπο” και έχει τις εξής ιδιότητες: όνομα, επώνυμο, και ηλικία. Σε μία οντοκεντρική γλώσσα, το αντικείμενο “πρόσωπο” θα επέτρεπε προσπέλαση στις ιδιότητες του ως εξής:

Person.name =”Bob”;

Person. Surname = ”Gates”;

Person.age = 54;

Με τον ίδιο τρόπο επιτρέπει και η Javascript την πρόσβαση στις ιδιότητες ενός αντικειμένου, μόνο που εφόσον αποτελεί πίνακα συσχετισμού, η πρόσβαση μπορεί να γίνει και ως εξής:

Person[“name”] = “Bob”;

Person[“surname”] = ”Gates”;

Person[“age”] = “54”;

Βλέπουμε ότι τα ονόματα των ιδιοτήτων αποτελούν κλειδιά αλφαριθμητικών στον πίνακα συσχετισμού. Κατά την εκτέλεση του προγράμματος είναι δυνατή η δυναμική προσθήκη και διαγραφή ιδιοτήτων σε ένα αντικείμενο.

Οι συναρτήσεις στην Javascriptείναι τύπου first class functions. Οι ίδιες οι συναρτήσεις είναι στην πραγματικότητα αντικείμενα. Εκτός από σώμα εκτελέσιμου κώδικα, ο οποίος μπορεί να καλεστεί ανά πάσα στιγμή, έχουν ιδιότητες όπως κάθε άλλο αντικείμενο στην Javascript. Επίσης επιτρέπεται η δήλωση συναρτήσεων μέσα σε σώματα άλλων συναρτήσεων (innerfunctions). Οι συναρτήσεις αυτές έχουν πρόσβαση στις μεταβλητές της εξωτερικής συνάρτησης ακόμα και αν αυτή έχει τελειώσει την εκτέλεσή της.

Η Javascript έχει τα prototypes αντί του μηχανισμού των κλάσεων και της κληρονομικότητας. Μπορούμε να εξομοιώσουμε πολλά στοιχεία του οντοκεντρικού προγραμματισμού χρησιμοποιώντας prototypesστην javascript. Για objectconstructors χρησιμοποιούνται απλές συναρτήσεις οι οποίες καλούνται με την λέξη κλειδί newσαν πρόθεμα, ώστε να δημιουργήσουν ένα νέο αντικείμενο. Η λέξη κλειδί thisστο σώμα μιας συνάρτησης η οποία καλείται σαν objectconstructor, αναφέρεται στο καινούργιο αντικείμενο που δημιουργήθηκε. Για να γίνει πιο κατανοητός ο μηχανισμός δημιουργίας αντικειμένων στην Javascript, παραθέτουμε το παρακάτω παράδειγμα:

Έστω ότι θέλουμε να δημιουργήσουμε ένα αντικείμενο που αναπαριστά το γεωμετρικό σχήμα παραλληλόγραμμο, αποθηκεύοντας πληροφορίες για το πλάτος και το μήκος και έχοντας μία μέθοδο που να υπολογίζει το εμβαδό του. Έστω το όνομα της κλάσης του αντικειμένου ότι είναι Rectangle. Στην Javascriptη δήλωση της κλάσης μπορεί να γίνει ώς εξής:

function Rectangle(w,h){

this.width = w;

this.height = h;

}

Η παραπάνω συνάρτηση δουλεύει ως objectconstructor. Δηλαδή όταν θέλουμε να δημιουργήσουμε ένα νέο παραλληλόγραμμο θα καλούμε για παράδειγμα:

myRectangle = newRectangle(100,200);

και θα δημιουργείται ένα αντικείμενο τύπου rectangle, με τιμές των ιδιοτήτων width=100, height=200. Αν θέλουμε να προσθέσουμε την μέθοδο getArea() που θα υπολογίζει το εμβαδό τότε θα πρέπει να χρησιμοποιήσουμε τα prototypes. Η δήλωση γίνεται ως εξής:

Rectangle.prototype.getArea = function()

{

*returnthis.width*this.height;*

}

Η κλήση της στο αντικείμενο myRectangleγίνεται ώς εξής:

myRectangle.getArea();

και επιστρέφει 2000 για το συγκεκριμένο παράδειγμα.

Γενικά αντιλαμβανόμαστε ότι τα αντικείμενα στην Javascriptπεριλαμβάνουν ιδιότητες οι οποίες έχουν συγκεκριμένες τιμές για το κάθε αντικείμενο ξεχωριστά, αλλά μπορούν να περιέχουν και ιδιότητες οι οποίες είναι ίδιες για όλα τα αντικείμενα. Οι ιδιότητες οι οποίες παραμένουν ίδιες για όλα τα αντικείμενα της javascript δηλώνονται στο prototype μέρος ενός αντικειμένου. Για τον λόγο αυτό και οι μέθοδοι οι οποίες στην ουσία αποτελούν μία συνάρτηση που εκτελείται με τον ίδιο τρόπο για κάθε αντικείμενο της κλάσης, δηλώνονται σαν συναρτήσεις στο prototypeμέρος του αντικειμένου.

Για να χρησιμοποιήσουμε στοιχεία (elements) της HTMLσελίδας μέσω του κώδικα Javascript, δίνουμε στα στοιχεία αυτά ένα id. Στον κώδικα javascriptμπορεί να γίνει αναφορά στο συγκεκριμένο στοιχείο εκτελώντας την εντολή document.getElementById(“id_name”).Στο επόμενο παράδειγμα (εικόνα 3.5) θα αλλάξουμε το περιεχόμενο του Htmlelementp, με χρήση κώδικα javascript.

Εικόνα 3.5-Παράδειγμα αλλαγής περιεχομένου Htmlelementμε χρήση Javascript.

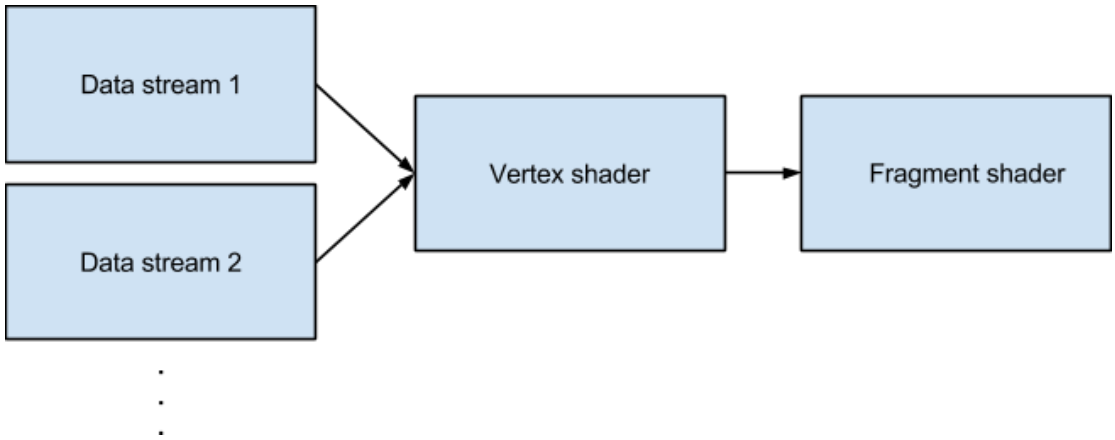
ΤαHtmlevents συμβαίνουν όταν κάτι γίνεται στα Htmlelements. Ένα Htmleventμπορεί να είναι κάτι που κάνει ο φυλλομετρητής, ή κάτι που κάνει ο χρήστης. Για παράδειγμα υπάρχει event όταν φορτωθεί η ιστοσελίδα, ‘η όταν κάτι γραφεί σε μία φόρμα, ή όταν πατηθεί ένα κουμπί της Html, όταν κινηθεί το ποντίκι, καθώς και όταν πατηθεί ένα πλήκτρο στο πληκτρολόγιο. Συνήθως όταν συμβαίνει ένα event, δεν θέλουμε να το χρησιμοποιήσουμε, αλλά όταν θελήσουμε, τότε η Javascriptμας επιτρέπει να τρέξουμε κάποιο κώδικα που έχει γραφεί όταν εντοπίζεται ένα event. Στην ουσία ενσωματώνουμε event listeners μέσω Javascriptστα Htmlelementsώστε να «πιάνουν» τα eventsπου πυροδοτούνται πάνω στο συγκεκριμένο στοιχείο. Αυτά λοιπόν καλούν την συνάρτηση που έχουμε φτιάξει ώστε να αξιοποιήσουμε τα evens.

Η χρήση της javascript στην εφαρμογή μας, έγκειται κυρίως στη χρήση της υπό ανάπτυξης Javascript βιβλιοθήκης, Three.js η οποία έχει δημιουργηθεί με σκοπό να διευκολύνει τους προγραμματιστές που θέλουν να εκμεταλλευτούν την WebGL στον φυλλομετρητή. Επίσης χρησιμοποιήθηκε για να μπορούμε να κάνουμε ασύγχρονες κλήσεις στη βάση (μέσω AJAJ - Asynchronous Javascript and JSON), προσθέτοντας αμεσότητα στην εφαρμογή μας και μείωση στο μέγεθος των δεδομένων που μεταφέρονταν κατά τη χρήση αυτής.

3.2.4 WebGL

Σε προηγούμενο κεφάλαιο κάναμε μία σύντομη αναφορά για τα πλεονεκτήματα της WebGL (Web Graprhics Library) και γιατί την επιλέξαμε ως βασική τεχνολογία στην διπλωματική αυτή. Η WebGL ουσιαστικά φέρνει τα τρισδιάστατα γραφικά στον φυλλομετρητή μας μέσω της HTML5. Όπως έχουμε αναφέρει η WebGL βασίζεται στην OpenGL ES 2.0 η οποία αποτελεί ένα υποσύνολο της OpenGL(Open Graphics Library) και παρέχει πρόσβαση στην κάρτα γραφικών. Η OpenGL,OpenGL ES και η WebGL όμως έχουν μία διαφορά σε σχέση με άλλα API γραφικών. Ηδιαφορά είναι ότι έχουν σαν βασικό δομικό στοιχείο για κάθε σκηνή τρισδιάστατη το τρίγωνο. Τα δεδομένα για πολλά τρίγωνα υπολογίζονται μία φορά αλλά σχεδιάζονται πολλές. Τα δεδομένα που αποστέλλονται στην κάρτα γραφικών (GPU) περιλαμβάνουν έννοιες όπως θέσεις κορυφών, χρώματα, υφές και άλλα.

Οι shaders είναι μικρά προγράμματα γραμμένα σε GLSL, η οποία είναι υψηλού επιπέδου γλώσσα βασισμένη στη C, που εκτελούνται απευθείας στην κάρτα γραφικών και προσδιορίζουν τις θέσεις κάθε τριγώνου καθώς και το χρώμα του κάθε ιχνοστοιχείου (pixel) στην οθόνη. Η WebGL χρησιμοποιεί ένα steam μοντέλο επεξεργασίας. Κάθε σημείο στον τρισδιάστατο χώρο έχει ένα η περισσότερα streamsαπό δεδομένα που σχετίζονται με αυτό όπως την θέση, το χρώμα την υφή και άλλα. Στην OpenGLκαθώς και την WebGLαυτά τα δεδομένα ονομάζονται vertexattributes, και επεξεργάζονται από τους VertexκαιFragmentshaders.(εικόνα 3.6)

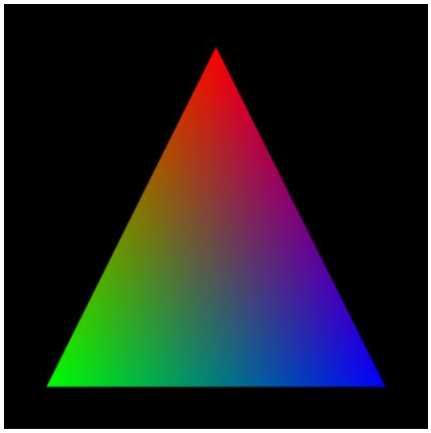


Εικόνα 3.6- Διαγραμματικά η επεξεργασία των VertexAttributes (Datastreams)

OVertexShaderυπολογίζει τα δεδομένα κάθε κορυφής κάθε τριγώνου, και στόχος του προγράμματος αυτού είναι να εξάγει την θέση όπου η κάθε κορυφή πρέπει να εμφανίζεται στο παράθυρο της οθόνης. Επιπλέον ο vertexshaderμπορεί να εξάγει πρόσθετες τιμές που ονομάζονται varying variables στον fragmentshader. Για κάθε τρίγωνο η κάρτα γραφικών υπολογίζει ποιά pixelστην οθόνη καλύπτονται από το τρίγωνο. Στην συνέχεια η κάρτα γραφικών τρέχει τον fragmentshader για κάθε ένα από αυτά τα pixels, ο οποίος υπολογίζει το χρώμα του εικονοστοιχείου.

3.2.4.1 Παράδειγμα χρήσης WebGL

Θα εξηγήσουμε ένα παράδειγμα χρήσης της WebGlγια την απεικόνιση ενός δισδιάστατου πολύχρωμου τριγώνου. Το αποτέλεσμα στον φυλλομετρητή φαίνεται στην παρακάτω εικόνα .(εικόνα 3.7)



Εικόνα 3.7- Αποτέλεσμα εφαρμογής του κώδικα του παραδείγματος.

Αρχικά γράφουμε τον κώδικα για τους shaders.

```
<script id="shader-vs" type="x-shader/x-vertex">

attribute vec3 positionAttr;
attribute vec4 colorAttr;

varying vec4 vColor;

void main(void) {
    gl_Position = vec4(positionAttr, 1.0);
    vColor = colorAttr;
}
</script>
```

Εικόνα 3.8- Παράδειγμα κώδικα vertexshader

Παρατηρώντας προσεχτικά τον παραπάνω κώδικα (εικόνα 3.8) βλέπουμε ότι το πρόγραμμά μας δίνει στην κάρτα γραφικών πίνακα τεσσάρων διαστάσεων για την θέση της κάθε κορυφής. Αυτό γίνεται καθαρά για ταχύτερους υπολογισμούς στην κάρτα γραφικών. Επιπλέον παρατηρούμε ότι χρησιμοποιούμε την μεταβλητή colorAttrκαι στον vertexshader, καθώς από αυτόν θα μεταβεί στον fragmentshaderγια τον υπολογισμό του χρώματος.Σε αυτό το παράδειγμα ο vertexshaderθα κληθεί μόλις τρεις φορές, καθώς τόσες είναι οι κορυφές του τριγώνου που θέλουμε να σχεδιάσουμε. Σε μία σύνηθες εφαρμογή ο vertexshaderκαλείται να υπολογίσει δεκάδες χιλιάδες τρίγωνα που θα ζωγραφιστούν στο canvas όλα μαζί.

```
<script id="shader-fs" type="x-shader/x-fragment">

precision mediump float;

varying vec4 vColor;
void main(void) {
    gl_FragColor = vColor;
}
</script>
```

Εικόνα 3.9- Παράδειγμα κώδικα υλοποίησης fragmentshader

Ο fragmentshader (εικόνα 3.9)όπως είπαμε καλείται για να υπολογιστεί το χρώμα κάθε pixelπου καλύπτεται από το τρίγωνο. Η μεταβλητή vColorπροέρχεται από ένα

σταθμισμένο συνδυασμόχρωμάτων που καθορίζονται στις τρείς κορυφές εισόδου. Με βάση την τοποθεσία του κάθε pixelεντός του τριγώνου, η κάρτα γραφικών συνδυάζει αυτόματα το χρώμα που καθορίζεται σε κάθε κορυφή, και έτσι αποδίδεται το χρώμα του συγκεκριμένου pixel. Η κλήση του fragmentshaderγίνεται τόσες φορές όσα είναι και τα pixelπου επικαλύπτονται από το τρίγωνο που σχεδιάζεται στην οθόνη.

Επόμενο βήμα για την υλοποίηση του παραδείγματος είναι να γίνει ο έλεγχος ότι ο υπολογιστής που θα τρέξει το παράδειγμα υποστηρίζει την WebGl καθώς και να γίνει η αρχικοποίηση της ώστε να μπορεί να τρέξει το πρόγραμμα με βάση την WebGl. (εικόνα 3.10)

```
var gl = null;
var contextNames = [ "webgl", "experimental-webgl" ];
for (var ii = 0; ii < contextNames.length; ++ii) {
    try {
        gl = canvas.getContext(contextNames[ii]);
        if (gl)
            break;
    } catch (e) {
    }
}
if (!gl) {
    alert("Could not initialise WebGL, sorry :-(");
}
```

Εικόνα 3.10- Κώδικας αρχικοποίησης της WebGl.

Στη συνέχεια αφού αρχικοποιήσαμε την WebGl και γράψαμε τους shaders, πρέπει να καλέσουμε τους shadersαυτούς ώστε να μπορεί να τους χρησιμοποιήσει η WebGl.

```
function getShader(gl, id) {
  var script = document.getElementById(id);
  var shader;
  if (script.type == "x-shader/x-vertex") {
    shader = gl.createShader(gl.VERTEX_SHADER);
  } else if (script.type == "x-shader/x-fragment") {
    shader = gl.createShader(gl.FRAGMENT_SHADER);
  }
  gl.shaderSource(shader, script.text);
  gl.compileShader(shader);
  if (!gl.getShaderParameter(shader, gl.COMPILE_STATUS)) {
    alert(gl.getShaderInfoLog(shader));
    return null;
  }

  return shader;
}
```

Εικόνα 3.11- Κώδικας κλήσης των shaders

Στον παραπάνω κώδικα (εικόνα 3.11) αρχικά δημιουργούμε ένα αντικείμενο για τον vertexκαι τον fragmentshader, καθορίζεται ο πηγαίος κώδικας τους, γίνεται η μεταγλώττιση του κώδικα και ελέγχεται αν όλα πήγαν καλά στην μεταγλώττιση(compile).

```
var program;
function initShaders() {
  program = gl.createProgram();
  var vertexShader = getShader(gl, "shader-vs");
  gl.attachShader(program, vertexShader);
  var fragmentShader = getShader(gl, "shader-fs");
  gl.attachShader(program, fragmentShader);
  gl.linkProgram(program);
  if (!gl.getProgramParameter(program, gl.LINK_STATUS))
    alert("Could not initialise shaders");
  gl.useProgram(program);
  program.positionAttr =
    gl.getAttribLocation(program, "positionAttr");
  gl.enableVertexAttribArray(program.positionAttr);
  program.colorAttr =
    gl.getAttribLocation(program, "colorAttr");
  gl.enableVertexAttribArray(program.colorAttr);
}
```

Εικόνα 3.12- Υλοποίηση του programobject

Επόμενο βήμα είναι να κατασκευαστεί ένα αντικείμενο-πρόγραμμα(programobject) (εικόνα 3.12). Αυτό συνδυάζει τον fragmentκαι vertexshaderκαι καθορίζει πως αποδίδεται ένα κομμάτι της γεωμετρίας του αντικειμένου που θέλουμε να εμφανίσουμε. Αρχικά φορτώνει τον vertexκαι fragmentshaderξεχωριστά και τους συνδέει με το πρόγραμμα, στην συνέχεια συνδέεται το πρόγραμμα και εκτελούνται οι απαραίτητοι έλεγχοι.

Αφού λοιπόν δημιουργήθηκαν όλα τα στοιχεία που είναι απαραίτητα για τον υπολογισμό και την απεικόνιση της γεωμετρίας μας, πρέπει να ορίσουμε την γεωμετρία και τα χρώματα που θα έχει. Χρησιμοποιούμε 3 κορυφές καθώς είναι τρίγωνο, και θέτουμε για αυτές και το αντίστοιχο χρώμα.(εικόνα 3.13)

```
var buffer;
function initGeometry() {
  buffer = gl.createBuffer();
  gl.bindBuffer(gl.ARRAY_BUFFER, buffer);
  // Interleave vertex positions and colors
  var vertexData = [
    // Vertex 1 position
    0.0,  0.8,  0.0,
    // Vertex 1 Color
    1.0, 0.0, 0.0, 1.0,
    // Vertex 2 position
    -0.8, -0.8,  0.0,
    // Vertex 2 color
    0.0, 1.0, 0.0, 1.0,
    // Vertex 3 position
    0.8, -0.8,  0.0,
    // Vertex 3 color
    0.0, 0.0, 1.0, 1.0
  ];
  gl.bufferData(gl.ARRAY_BUFFER,
    new Float32Array(vertexData), gl.STATIC_DRAW);
}
```

Εικόνα 3.13- Ορισμός της γεωμετρίας και των χρωμάτων του τριγώνου.

Σε αυτό το σημείο έχουμε όλα τα κομμάτια που χρειαζόμαστε , αρκεί να τα ενώσουμε για να εμφανιστεί στον φυλλομετρητή το αποτέλεσμα που θέλουμε. Στην

παρακάτω συνάρτηση -drawscene()- αρχικά «καθαρίζουμε» το canvaselement, ορίζουμε τα vertexattributeskαι «ζωγραφίζουμε» στο canvastην σκηνή.(εικόνα 3.14)

```
1  function drawScene() {
2    gl.viewport(0, 0, canvas.width, canvas.height);
3    gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);
4
5    gl.bindBuffer(gl.ARRAY_BUFFER, buffer);
6
7    // There are 7 floating-point values per vertex
8    var stride = 7 * Float32Array.BYTES_PER_ELEMENT;
9
10   // Set up position stream
11   gl.vertexAttribPointer(program.positionAttr,
12     3, gl.FLOAT, false, stride, 0);
13   // Set up color stream
14   gl.vertexAttribPointer(program.colorAttr,
15     4, gl.FLOAT, false, stride,
16     3 * Float32Array.BYTES_PER_ELEMENT);
17
18   gl.drawArrays(gl.TRIANGLES, 0, 3);
19 }
```

Εικόνα 3.14 – Υλοποίηση της drawscenesυνάρτησης στην WebGl

Τέλος είναι έτοιμο το τρίγωνο να εμφανιστεί στον φυλλομετρητή, τρέχοντας την εφαρμογή. (εικόνα 3.7)

Η WebGL είναι χαμηλού επιπέδου και για αυτό τον λόγο υπάρχουν βιβλιοθήκες οι οποίες διευκολύνουν την χρήση τρισδιάστατων γραφικών στον φυλλομετρητή μέσω WebGL. Οι βιβλιοθήκες αυτές παρέχουν στον προγραμματιστή την δυνατότητα να γράψει δομημένη javascript, χωρίς να χρειαστεί να γράψει shaders και να ορίσει το πρόγραμμα με το οποίο θα τρέχουν.

3.2.5 Three.js

Η Three.js είναι μία ελαφριά βιβλιοθήκη της γλώσσας Javascript και χρησιμοποιείται για την δημιουργία και την απεικόνιση τρισδιάστατων γραφικών στον φυλλομετρητή. Χρησιμοποιεί την τεχνολογία WebGL που περιγράψαμε σε προηγούμενο κεφάλαιο, και η οποία είναι η επιλογή μας για την απεικόνιση τρισδιάστατων γραφικών. Καθώς χρησιμοποιεί την WebGL επιτρέπει την δημιουργία τρισδιάστατων γραφικών επιταχυνόμενων από την κάρτα γραφικών με την γλώσσα Javascript.Η Three.js περιλαμβάνει χαρακτηριστικά όπως κάμερες, φωτισμό, τρισδιάστατα γεωμετρικά σχήματα, φορτωτές δεδομένων(loaders) για εικόνες και διάφορους τύπους αρχείων όπως τα .obj, που μας παρείχαν οι φοιτητές της αρχιτεκτονικής. Ο πιο εύκολος τρόπος να εξηγήσουμε την χρήση της three.jsείναι να εξετάσουμε ένα παράδειγμα, καθώς και οι ίδιοι οι δημιουργοί της το προτείνουν διότι τα έγγραφα με τις οδηγίες χρήσης είναι πολύ «φτωχά» ακόμη. Μέσω του παραδείγματος θα δείξουμε και την διαφορά της στην χρήση σε σχέση με την WebGLαυτούσια χωρίς χρήση βιβλιοθήκης.

3.2.5.1 Παράδειγμα χρήσηςThree.js

Στο παράδειγμα αυτό θα δείξουμε πως μπορούμε να δημιουργήσουμε ένα τρισδιάστατο κύβο ο οποίος περιστρέφεται γύρω από τον χκαι γάξονα του. με χρήση των συναρτήσεων που παρέχονται από την three.js. Αρχικά εισάγουμε στον κώδικα της HTML5 που θέλουμε να εμφανίσουμε τον κύβο, την βιβλιοθήκη three.min.js (εικόνα 3.15). Ηβιβλιοθήκη αυτή είναι συμπιεσμένη για λόγους ταχύτητας μεταφοράς των δεδομένων, και δεν μπορούμε να διαβάσουμε τον κώδικά της. Όμως υπάρχει και

η Three.js η οποία είναι η ανεπτυγμένη της μορφή και μπορούμε να την επεξεργαστούμε αν θέλουμε να αλλάξουμε κάτι.

```
<html>
    <head>
        <title>My first Three.js app</title>
        <style>
            body { margin: 0; }
            canvas { width: 100%; height: 100% }
        </style>
    </head>
    <body>
        <script src="js/three.min.js"></script>
    </body>
</html>
```

Εικόνα 3.15- HTMLκώδικας, για την δημιουργία τρισδιάστατης σκηνής με χρήση Three.js

Για να μπορέσουμε να εμφανίσουμε το οτιδήποτε στον φυλλομετρητή μας χρησιμοποιώντας την βιβλιοθήκη three.js, χρειαζόμαστε 3 βασικά «αντικείμενα». Μια σκηνή(scene), μία κάμερα και έναν rendererώστε να μπορέσει να αποτυπωθεί-ζωγραφιστεί η σκηνή.

```
var scene = new THREE.Scene();
var camera = new THREE.PerspectiveCamera( 75, window.innerWidth / window.innerHeight, 0.1, 1000 );

var renderer = new THREE.WebGLRenderer();
renderer.setSize( window.innerWidth, window.innerHeight );
document.body.appendChild( renderer.domElement );
```

Εικόνα 3.16- Αρχικοποίηση της σκηνής της κάμερας και του renderer.

Στον κώδικα Javascriptπαραπάνω,(εικόνα 3.16), αρχικοποιούμε την κάμερα, την σκηνή και τον renderer. Η Three.jsμας δίνει αρκετές επιλογές για την κάμερα, και στο συγκεκριμένο παράδειγμα χρησιμοποιήσαμε την Perspective. Δέχεται σαν πρώτο όρισμα το οπτικό πεδίο το οποίο ορίσαμε στις 75 μοίρες. Σαν δεύτερο όρισμα δέχεται την αναλογία των διαστάσεων (aspectratio), η οποία σχεδόν πάντα για να έχουμε το επιθυμητό αποτέλεσμα πρέπει να ισούται με τον λόγο του πλάτους του HTMLcanvasδιαιρούμενο από το ύψος του, ειδάλλως το αποτέλεσμα θα είναι σαν να συμπιέζονται τα αντικείμενα της σκηνής. Τα δύο άλλα ορίσματα αφορούν στο πόσο κοντά και πόσο μακριά από την κάμερα δεν θα αποτυπώνονται τα αντικείμενα που έχει η σκηνή. Το επόμενο που ορίζουμε, είναι ο renderer, όπου στην προκειμένη περίπτωση χρησιμοποιούμε WebGLRenderer, αν και η three.jsδίνει την δυνατότητα να χρησιμοποιηθούν και άλλοι, όχι τόσο σύγχρονοι, κυρίως για χρήστες που δεν έχουν σύγχρονους φυλλομετρητές. Τα ορίσματα που δέχεται είναι το ύψος και το πλάτος που θέλουμε να χρησιμοποιήσει. Τέλος, δημιουργήσαμε μέσω Javascriptto canvaselement.

Αυτό που μας απομένει είναι να δημιουργήσουμε τον κύβο και να τον εμφανίσουμε στην σκηνή.

```
1 var geometry = new THREE.BoxGeometry( 1, 1, 1 );
2 var material = new THREE.MeshBasicMaterial( { color: 0x00ff00 } );
3 var cube = new THREE.Mesh( geometry, material );
4 scene.add( cube );
5
6 camera.position.z = 5;
```

Εικόνα 3.17- κώδικας για την δημιουργία κύβου με την χρήση Three.js

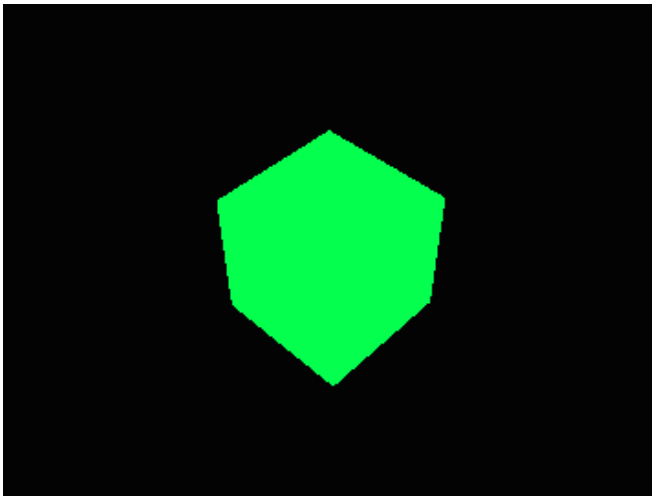
Για να δημιουργήσουμε ένα κύβο χρειαζόμαστε ένα BoxGeometry αντικείμενο, το οποίο είναι της βιβλιοθήκης που εξετάζουμε, και περιέχει όλα τα σημεία(κορυφές), και τις επιφάνειες που δημιουργούνται από αυτά. Επιπλέον χρειαζόμαστε ένα υλικό για να τον χρωματίσουμε, υπάρχουν πολλά είδη υλικών στην Three.js, αλλά στο παράδειγμα μας χρησιμοποιήσαμε το MeshBasicMaterial. Όλα τα υλικά παίρνουν σαν ορίσματα ένα πλήθος από επιλογές όπως το χρώμα, το αν αλληλοεπιδρά με τον υπόλοιπο «κόσμο», το αν χρησιμοποιεί κάποια εικόνα, και πολλά άλλα. Στο προκειμένο παράδειγμα ορίσαμε το χρώμα του υλικού μόνο, και το 0x00ff00 στο δεκαεξαδικό σύστημα μας δίνει το πράσινο χρώμα. Το τρίτο που χρειαζόμαστε, είναι ένα Mesh, το οποίο ουσιαστικά είναι ένα αντικείμενο που παίρνει σαν όρισμα την γεωμετρία που ορίσαμε παραπάνω και της εφαρμόζει το υλικό που ορίσαμε. Τέλος εισήγαμε τον κύβο μας στην σκηνή, και τον μετατοπίσαμε στον άξονα z (βάθος) καθώς εξ αρχής κάθε αντικείμενο μας τοποθετείτε στην σκηνή στις συντεταγμένες (0,0,0), και καθώς δεν μετατοπίσαμε την κάμερα, αν αφήναμε τον κύβο στο ίδιο σημείο, θα βρισκόταν η κάμερα μέσα στον κύβο.

Παρότι έχουμε εισάγει όμως τον κύβο στην σκηνή, αν τρέξουμε την εφαρμογή δεν θα δούμε τίποτα σαν αποτέλεσμα. Αυτό συμβαίνει διότι δεν προβάλουμε τίποτα ακόμη. Με τον παρακάτω κώδικα δημιουργείται ένας επαναληπτικόςβρόχος ο οποίος καλεί τον rendererva «ζωγραφίζει»την σκηνή 60 φορές το δευτερόλεπτο(60 framespersecond). Τέλος μέσα στην συνάρτηση αυτή αυξάνουμε την περιστροφή του κύβου στον x και yάξονα περιστροφής του, και αυτό έχει σαν αποτέλεσμα να περιστρέφεται συνέχεια, αφού η συνάρτηση αυτή καλείται 60 φορές ανά δευτερόλεπτο.(εικόνα 3.18)

```
1 function render() {
2     requestAnimationFrame( render );
3
4     cube.rotation.x += 0.1;
5     cube.rotation.y += 0.1;
6
7     renderer.render( scene, camera );
8 }
9 render();
```

Εικόνα 3.18- Υλοποίηση της συνάρτησης render, three.js

Βλέπουμε λοιπόν ένα τρισδιάστατο αποτέλεσμα με πολύ λίγες γραμμές κώδικα Javascript (εικόνα 3.19), και αρκετά πιο φιλικό προς τους προγραμματιστές από ότι η χαμηλού επιπέδου WebGL.



Εικόνα 3.19-Περιστρεφόμενος κύβος στον browser με την χρήση της Three.js

3.3 Πλευρά Διακομιστή (Server side)

Η πιο θεμελιώδης πτυχή της υλοποίησης είναι ο καθορισμός του περιβάλλοντος μέσα στον οποίο θα αναπτυχθεί, θα εδρεύει και θα εκτελείται η εφαρμογή. Επιλέξαμε να υλοποιήσουμε σε γλώσσα Java το τμήμα της εφαρμογής που βρίσκεται στον διακομιστή ιστού, επομένως επιλέξαμε να χρησιμοποιήσουμε τον Apache Server, και συγκεκριμένα τον Apache Tomcat, ο οποίος χρησιμοποιείται για την ανάπτυξη Java Servlet.

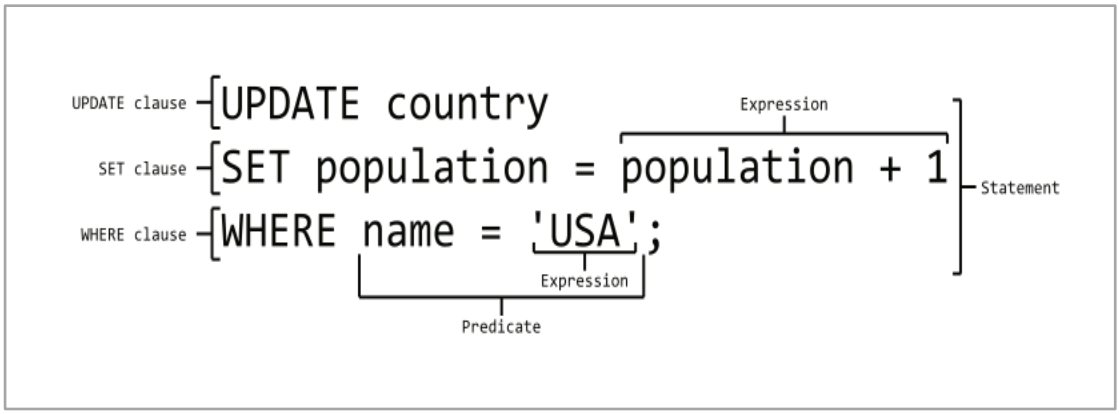
3.3.1 Servlets

Τα servlets είναι μία τεχνολογία που χρησιμοποιείται για την λήψη και την απάντηση σε αιτήματα (requests) από τον πελάτη χρήστη (client), συνήθως μέσω του πρωτοκόλλου HTTP. Πρόκειται για μικρά κομμάτια κώδικα που εκτελούνται στην πλευρά του διακομιστή. Κάθε servlet έχει τον ακόλουθο κύκλο ζωής: αρχικά κατασκευάζεται και αρχικοποιείται. Στη συνέχεια, αφού κληθεί, διαχειρίζεται το αίτημα που δέχεται από τον client και στέλνει μία απάντηση ώς επιστροφή. Τέλος, αφού ολοκληρώσει την εργασία του, καταστρέφεται. Στην εφαρμογή μας η τεχνολογία αυτή χρησιμοποιήθηκε για να απαντά στα αιτήματα που υποβάλλει ο client. Για παράδειγμα αν ο χρήστης επιλέξει να δεί κάποιο μνημείο, επιλέγοντας από τον φυλλομετρητή του μία πινέζα (pin), ένα servlet είναι υπεύθυνο για να αποκριθεί στο αίτημα και να επιστρέψει στο χρήστη πληροφορίες για το αν όλα πήγαν καλά, δηλαδή στο συγκεκριμένο παράδειγμα θα επιστραφεί το μοντέλο που είχε επιλεγεί.

3.3.2 SQL

Η SQL (Structured Query Language) είναι μία γλώσσα προγραμματισμού για βάσεις δεδομένων, που σχεδιάστηκε για την διαχείριση δεδομένων σε ένα σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων, η οποία αρχικά βασίστηκε στην σχεσιακή άλγεβρα. Η γλώσσα περιλαμβάνει δυνατότητες ανάκτησης και ενημέρωσης δεδομένων, δημιουργίας και τροποποίησης σχημάτων και σχεσιακών πινάκων, αλλά και ελέγχου πρόσβασης στα δεδομένα. Η SQL ήταν από τις πρώτες γλώσσες για το σχεσιακό μοντέλο και έγινε η πιο ευρέως χρησιμοποιούμενη γλώσσα για τις σχεσιακές βάσεις δεδομένων. Η γλώσσα SQL υποδιαιρείται σε διάφορα γλωσσικά στοιχεία, που περιλαμβάνουν(εικόνα 3.20):

- Clauses, οι οποίες είναι σε μερικές περιπτώσεις προαιρετικές, αλλά απαραίτητα συστατικά των δηλώσεων και ερωτήσεων.
- Expressions που μπορούν να παραγάγουν είτε τις κλιμακωτές τιμές είτε πίνακες που αποτελούνται από στήλες και σειρές στοιχείων.
- Predicates που διευκρινίζουν τους όρους που μπορούν να αξιολογηθούν σαν σωστό ή λάθος.
- Queries που ανακτούν τα στοιχεία βασισμένες σε ειδικά κριτήρια.
- Statements που μπορούν να έχουν μια επίδραση στα σχήματα και τα στοιχεία, ή που μπορούν να ελέγξουν τη ροή του προγράμματος και τις συνδέσεις από άλλα προγράμματα.
- Το κενό αγνοείται γενικά στις Statements και τις QueriesSQL. Ένα κενό είναι όμως απαραίτητο για να ξεχωρίζει Statements όπως και στην κανονική γραφή κειμένων.



Εικόνα 3.20- Γλωσσικά στοιχεία σε ένα SQLstatement.

Για παράδειγμα εάν έχουμε ένα πίνακα”Students” στην βάση δεδομένων μας η οποία περιέχει τους φοιτητές του πολυτεχνείου και έχει τα στοιχεία που μας δείχνει ο παρακάτω πίνακας.

A_M	firstName	lastName	grade	department
200403091	Dimitris	Dimitriou	8.5	ECE
200602014	Lambros	Konstantinou	9.1	ARCH
201002093	Anastasios	Petropoulos	8.4	ARCH
2007030162	Kostantinos	Loi	9.2	ECE

Αν θέλουμε τώρα να επιλέξουμε και να ανακτήσουμε τα στοιχεία των φοιτητών οι οποίοι ανήκουν στο τμήμα “ECE” και έχουν βαθμό μεγαλύτερο του 9.0, τότε θα χρειαστεί να εκτελέσουμε το παρακάτω επερώτημα (query). (εικόνα 3.21)

```
1 SELECT *
2 FROM student
3 WHERE grade > 9.0
4 AND department = 'ECE';
```

Εικόνα 3.21- Παράδειγμα SQL query

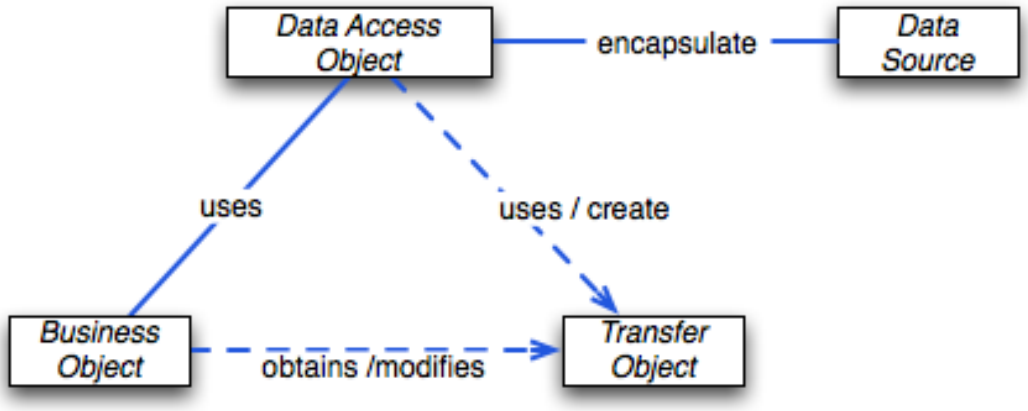
Το query αυτό έχει σαν αποτέλεσμα το παρακάτω πίνακα, ο οποίος είναι και η απάντηση στο ερώτημα μας:

A_M	firstName	lastName	grade	department
2007030162	Kostantinos	Loi	9.2	ECE

Το σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων που αποφασίσαμε να χρησιμοποιήσουμε είναι το MySQL. Εγκαταστήσαμε τον MySQLserver και σχεδιάσαμε την βάση μας σε αυτόν μέσω του γραφικού εργαλείου που παρέχεται, το MySQLWorkbench. Μέσω του γραφικού περιβάλλοντος δημιουργήσαμε την βάση δεδομένων μας την οποία συνδέσαμε με την εφαρμογή μας με την χρήση της βιβλιοθήκης JDBCπου παρέχεται.

3.3.3 Data Access Object (DAO)

Το μοτίβο Data Access Object διευκολύνει την πρόσβαση σε μία πηγή δεδομένων, στην περίπτωσή μας η πηγή αύτη είναι η βάση δεδομένων, για την αποθήκευση και ανάκτηση της πληροφορίας. Παρέχει όλους εκείνους τους μηχανισμούς (συναρτήσεις αποθήκευσης και ανάκτησης δεδομένων) οι οποίοι μας χρειάζονται για την πρόσβαση στην βάση δεδομένων. Το DAO αποκρύπτει πλήρως τις λεπτομέρειες υλοποίησης της πηγής δεδομένων από τους χρήστες της εφαρμογής(clients). Σημαντικό πλεονέκτημα της συγκεκριμένης τεχνολογίας είναι ότι διαχωρίζει το business logic επίπεδο της εφαρμογής από το persistent logic. Έτσι σε περίπτωση που μελλοντικά θέλουμε να κάνουμε μία αλλαγή στην πηγή δεδομένων οι clients δεν θα επηρεαστούν. Ουσιαστικά η δουλειά του DAO είναι να λειτουργεί σαν μεσολαβητής ανάμεσα στο business logic και την πηγή δεδομένων.(εικόνα 3.22)



Εικόνα 3.22- Διάγραμμα κλάσεων που αναπαριστά τις συσχετίσεις για το πρότυπο DAO.

Business Object: είναι το αντικείμενο που χρειάζεται να έχει πρόσβαση στην πηγή δεδομένων για την ανάκτηση και την αποθήκευση των δεδομένων.

Data Access Object: περιγράφει το επίπεδο υλοποίησης της πρόσβασης στα δεδομένα έτσι ώστε να επιτρέπει στο business object να έχει πρόσβαση στα δεδομένα μας.

Data Source: αντιπροσωπεύει την πηγή των δεδομένων της εφαρμογής. Μπορεί να είναι μία βάση δεδομένων όπως RDBMS,XML repository.

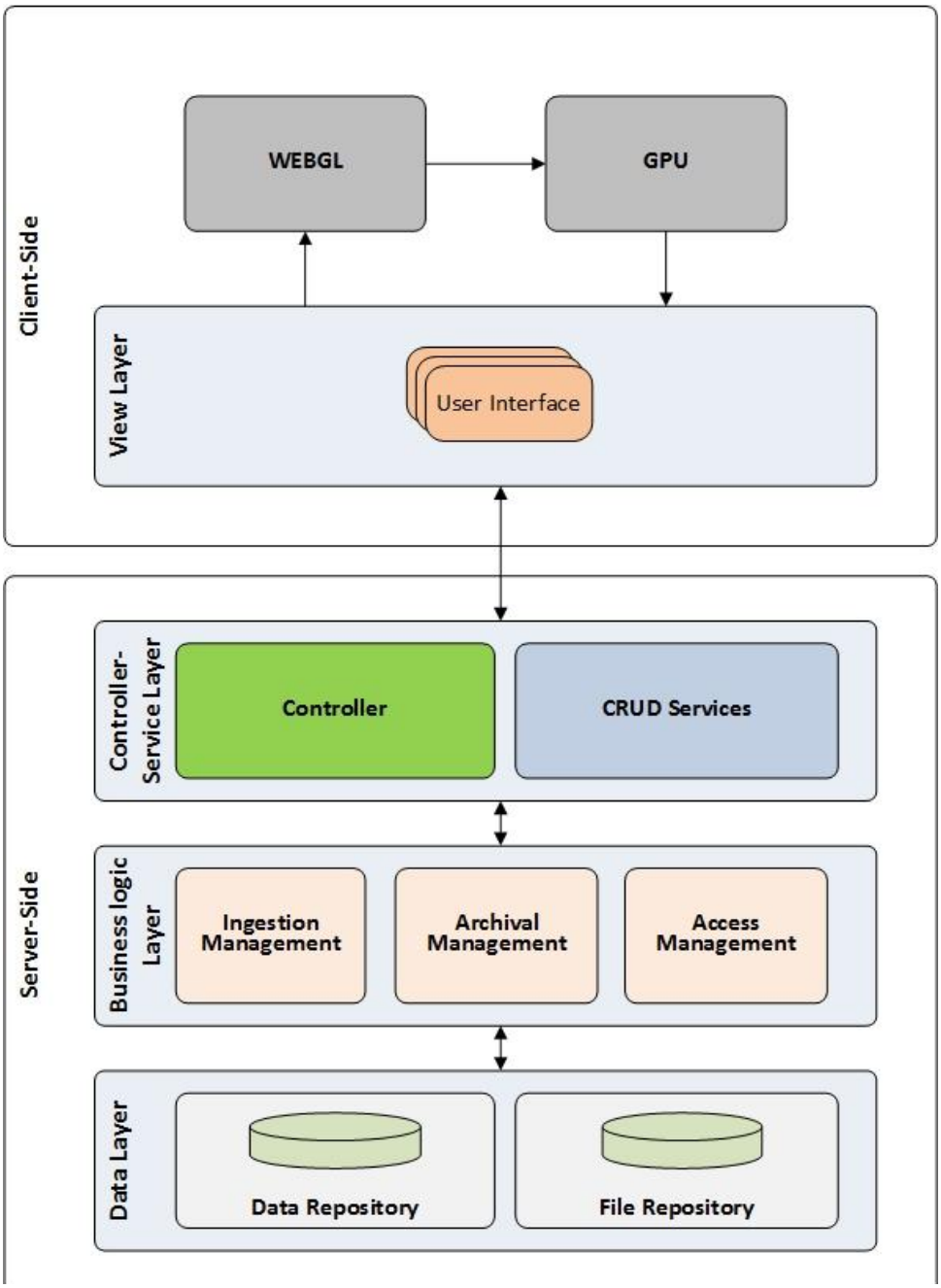
Transfer Object: χρησιμοποιείται για την μεταφορά των δεδομένων. Ένα data access object χρησιμοποιεί ένα transfer object για να επιστρέψει τα δεδομένα στον client.

3.4 Αρχιτεκτονική της Εφαρμογής

Η αρχιτεκτονική της εφαρμογής μας όπως βλέπουμε και στην εικονα-3.23 χωρίζεται σε ευρύτερα επίπεδα, στην πλευρά του πελάτη(client-side) και στην πλευρά του διακομιστή (server-side). Το κάθε ένα επίπεδο από αυτά εμπεριέχει επιμέρους επίπεδα και τμήματα τα οποία θα εξηγήσουμε με την σειρά.

3.4.1 Πλευρά Πελάτη (client-side)

Η πλευρά του πελάτη της εφαρμογής είναι υπεύθυνη για την αλληλεπίδραση με τον χρήστη. Όλες οι ενέργειες που εκτελούνται από ένα χρήστη αντιμετωπίζονται από την πλευρά του πελάτη η οποία αναλαμβάνει την παρουσίαση των πληροφοριών καθώς και την επικοινωνία με τον διακομιστή. Στην εφαρμογή μας, στην πλευρά του πελάτη, έχουμε ένα επίπεδο της εφαρμογής το οποίο έχουμε υλοποίηση το οποίο ονομάζεται ViewLayer. Αυτό είναι υπεύθυνο να εμφανίζει στον χρήστη τις πληροφορίες που έχει ανακτήσει από τον διακομιστή και επιπλέον μπορεί μέσω αυτού του επιπέδου το οποίο εμπεριέχει την γραφική διεπαφή της εφαρμογής να αλληλοεπιδράσει με τον διακομιστή. Στην πλευρά του πελάτη όπως βλέπουμε και στο διάγραμμα, το viewlayer επικοινωνεί μέσω της WebGlμε την κάρτα γραφικών, και έτσι εμφανίζονται στο viewlayerτα τρισδιάστατα γραφικά.



Εικόνα 3.23- Αρχιτεκτονική της εφαρμογής διαγραμματικά.

3.4.2 Πλευρά Διακομιστή (Server-side)

Η πλευρά του διακομιστή, ακολουθεί μία πολυεπίπεδη αρχιτεκτονική που αποτελείται από τρία βασικά επίπεδα. Τοεπίπεδοcontroller-service, τοBusiness Logic καιτοData layer. Ηεπιλογή πολυεπίπεδης αρχιτεκτονικής έχει πολλά πλεονεκτήματα, καθώς διευκολύνει στην συντήρηση της εφαρμογής, επιτρέπει την επαναχρησιμοποίηση οποιουδήποτε επιπέδου, βοηθά στην επεκτασιμότητα και στην ασφάλεια της εφαρμογής. Ουσιαστικά, αν στο μέλλον θέλουμε να κάνουμε κάποια διορθωτική ή επεκτατική αλλαγή, δεν χρειάζεται να αλλάξουμε όλη την εφαρμογή , παρά μόνο να επεκτείνουμε το κομμάτι του επιπέδου που θέλουμε.

3.4.2.1 Controller-service Layer

Το επίπεδο αυτό περιέχει τονControllerτης εφαρμογής και τα CRUDServices. OControllereίναι υπεύθυνος για την επεξεργασία και την απόκριση σε γεγονότα, όπως για παράδειγμα οι ενέργειες του χρήστη, και αποκρίνονται σε αυτά αλλάζοντας τα δεδομένα της εφαρμογής και τα views. Τα Services ελέγχουν ουσιαστικά την επικοινωνία μεταξύ του πελάτη (client) και του διακομιστή (server), παρέχοντας ένα σύνολο από υπηρεσίες(Services) προς τα στοιχεία της πλευράς του πελάτη. Αυτές οι υπηρεσίες αποτελούν ουσιαστικά το ενδιάμεσο λογισμικό το οποίο είναι υπεύθυνο για την απόκρυψη του BusinessLogic επιπέδου από τον πελάτη. Οι βασικές υπηρεσίες του συστήματος μας είναι CRUDServices (Create, Retrieve, Update, Delete), δηλαδή όπως λέει και το όνομά τους, χρησιμοποιούνται για την δημιουργία, την ανάκτηση, την ενημέρωση και την διαγραφή δεδομένων που αλληλοεπιδρά ο χρήστης, όπως στην περίπτωση της εφαρμογής μας ενός τρισδιάστατου μοντέλου, ή των multimediaαντικειμένων που συνδέονται με αυτό.

3.4.2.2 Business Logic Layer

BusinessLogicστην πληροφορική είναι το τμήμα του προγράμματος το οποίο κωδικοποιεί τους κανόνες για την διαχείριση δεδομένων όπως την δημιουργία την εμφάνιση, την αποθήκευση και την αλλαγή τους. Σε αυτό το επίπεδο ουσιαστικά διαχειριζόμαστε την πληροφορία από το datalayerκαι το controller-servicelayer και διαχωρίζει τα δύο αυτά επίπεδα.

- Το IngestionManagementδιαχειρίζεται το uploadingτων αρχείων στην εφαρμογή.
- Το ArchivalManagementείναι υπεύθυνο για την αποθήκευση των δεδομένων, και των αρχείων στην βάση δεδομένων και το fileSystemτου διακομιστή.
- Το AccessManagement, διαχειρίζεται την πρόσβαση στην πληροφορία που είναι αποθηκευμένη τόσο στην βάση δεδομένων όσο και στο filesystem.

3.5 Εργαλεία Προγραμματισμού

3.5.1 NetBeans IDE 8.0.1

Το NetBeans IDE είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης που χρησιμοποιείται από τους προγραμματιστές για την ανάπτυξη εφαρμογών. Είναι ουσιαστικά ένα εργαλείο που παρέχετε στους προγραμματιστές για να γράψουν, να κάνουν μεταγλώττιση (compile), και εντοπισμό σφαλμάτων(debug) στην εφαρμογή που αναπτύσσουν. Είναι ένα ελεύθερο προϊόν, χωρίς περιορισμούς στον τρόπο χρήσης του, και παρέχεται και μεγάλος αριθμός από υπομονάδες για την επέκταση της λειτουργικότητας του, ανάλογα με τις προτιμήσεις του χρήστη. Στο περιβάλλον αυτό δημιουργήσαμε την εφαρμογή μας, και γράψαμε τον κώδικα Javaπου ήταν απαραίτητο για να λειτουργεί η εφαρμογή μας στον server και να συνδέεται με τον clinet. Επιπλέον σε αυτό υλοποιήσαμε και τον κώδικα που βλέπει ο client στον φυλλομετρητή του, δηλαδή τον κώδικα Html, css, και javascript.

3.5.2 MySQL Connector

Η MySQL είναι όπως έχουμε αναφέρει και παραπάνω ένα σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων, που επιτρέπει την σύνδεση της βάσης με εφαρμογές που έχουν δημιουργηθεί χρησιμοποιώντας τη γλώσσα προγραμματισμού Java μέσω του JDBC(Java to DataBase Connectivity) οδηγού, ο οποίος ονομάζεται MySQL Connector.

3.5.3 MySQL Workbench

Για την διαχείριση και κατασκευή της βάσης δεδομένων μας, χρησιμοποιήσαμε το MySQL Workbench, το οποίο είναι ένα οπτικό εργαλείο σχεδιασμού βάσεων δεδομένων που ενσωματώνει την ανάπτυξη, την διαχείριση, την δημιουργία και την συντήρηση σε ένα ενιαίο ολοκληρωμένο περιβάλλον ανάπτυξης βάσης δεδομένων MySQL.

4 Ανάλυση απαιτήσεων

4.1 Εισαγωγή

Στο κεφάλαιο αυτό παρουσιάζονται όλες οι αποφάσεις που πάρθηκαν για την υλοποίηση της εφαρμογής. Πιο συγκεκριμένα θα παρουσιαστούν οι λειτουργικές απαιτήσεις της εφαρμογής μας, θα γίνει αναλυτική περιγραφή της εφαρμογής, θα παρουσιαστούν οι διαφορετικοί τύποι χρηστών που χρησιμοποιούν την εφαρμογή, καθώς και ανάλυση των περιπτώσεων χρήσης. Τέλος θα παρουσιαστεί ο σχεδιασμός της γραφικής διεπαφής της εφαρμογής.

4.2 Περιγραφή εφαρμογής

Η εφαρμογή που υλοποιήσαμε για την διπλωματική αυτή εργασία, όπως έχουμε αναφέρει και στην εισαγωγή, αποτελεί ιδέα του κ.Παρθένιου η οποία χρηματοδοτήθηκε από την εταιρία CytaHellasκαι βρίσκεται υπό την αιγίδα της νομαρχίας Κρήτης. Στην υλοποίηση αυτής της ιδέας, πολύ σημαντικό ρόλο είχε όλη η ομάδα η οποία εργάστηκε για την πραγματοποίηση της, και αποτελείται από φοιτητές και καθηγητές της αρχιτεκτονικής, που πήραν τις αποφάσεις για τον γραφικό σχεδιασμό των τρισδιάστατων μοντέλων. Η καινοτομία της ιδέας για την υλοποίηση της εφαρμογής έγγυται στη δυνατότητα που δίνεται στον χρήστη να περιηγηθεί μέσα από τον προσωπικό του υπολογιστή, και συγκεκριμένα μέσω του φυλλομετρητή του, στα μνημεία της Κρήτης και να παρατηρήσει την εξέλιξή τους κατά την διάρκεια επτά ιστορικών περιόδων. Ο χρήστης μέσω της εφαρμογής μπορεί να «επισκεφτεί» τα μνημεία μέσω του φυλλομετρητή του, και να γνωρίσει τον τρόπο με τον οποίο εξελίχτηκαν στην πάροδο του χρόνου και να πάρει πληροφορίες για τα μνημεία πριν τα επισκεφτεί. Μέσω των τεχνολογιών που αναλύθηκαν στα προηγούμενα κεφάλαια, δίνουμε την δυνατότητα στον χρήστη να γνωρίσει πως είναι τα μνημεία της Κρήτης όχι μέσω φωτογραφιών και κειμένου πληροφοριών μόνο, αλλά ως τρισδιάστατο μοντέλο με το οποίο μπορούν να αλληλοεπιδράσουν σε πραγματικό χρόνο. Υπάρχουν πέντε επίπεδα λεπτομέρειας για κάθε μνημείο, και αυτό δίνει την δυνατότητα να εμφανίζεται το εκάστοτε μνημείο από το επίπεδο της περιφέρειας της Κρήτης μέχρι και το επίπεδο κτηρίου. Επιπλέον βασικό στοιχείο της εφαρμογής είναι οι επτά βασικές χρονικέςπερίοδοι μέσω των οποίων ο χρήστης μπορεί να ενημερωθεί για την εξέλιξη των μνημείων σε αυτές. Η ιδέα, καθώς και η παραδοτέα εφαρμογή στην εταιρία CytaHellas, αφορούσε στην υλοποίηση 15 συνολικά μνημείων της Κρήτης σε όλα τα επίπεδα λεπτομέρειας, και σε 70 μνημείων σημειακού χαρακτήρα, στο επίπεδο της Περιφέρειας, δηλαδή στο πιο αφαιρετικό επίπεδο. Τα 15 μνημεία πλήρους λειτουργικότητας στην εφαρμογή, επιλέχθηκαν από την ομάδα της αρχιτεκτονικής, σύμφωνα με τα ιστορικά και αρχιτεκτονικά στοιχεία στα οποία είχαν πρόσβαση, καθώς και με βάση την «ιστορικότητας» τους. Μετά από πολλές συναντήσεις ολόκληρης της ομάδας, και έκφρασης ιδεών όλων των μελών, πάρθηκαν οι απαραίτητες αποφάσεις για τον τρόπο με τον οποίο υλοποιήθηκε η τελική εφαρμογή. Οι αποφάσεις αυτές αφορούσαν τον προσδιορισμό των επιπέδων λεπτομέρειας και στον τρόπο με τον οποίο παρουσιάζονται τα μνημεία σε κάθε επίπεδο. Στις επιμέρους πληροφορίες που θα παρέχονται για κάθε μνημείο, και τον τρόπο εμφάνισης τους, καθώς και στην διαδραστικότητα που παρέχεται στον χρήστη. Επιπλέον δόθηκαν στους υπεύθυνους σχεδιασμού των τρισδιάστατων μοντέλων βασικές προδιαγραφές που έπρεπε να ακολουθούν για να είναι εφικτή η

απεικόνιση των τρισδιάστατων μοντέλων στον φυλλομετρητή. Τα επίπεδα λεπτομέρειας της εφαρμογής ουσιαστικά αφορούν σε ένα είδος εστίασης στην επιμέρους πληροφορία που παρέχει το κάθε επίπεδο.

Τα πέντε βασικά επίπεδα της εφαρμογής είναι:

i. Κρήτη- (Crete)
ii. Νομός- (Prefecture)
iii. Οικισμός- (Region)
iv. Κτηριακό συγκροτήματα- (Complex)
v. Μνημείο – (Monument)

Στο πρώτο επίπεδο παρουσιάζεται το τρισδιάστατο μοντέλο της Κρήτης, το οποίο διαχωρίζεται σε Νομούς. Ο χρήστης επιλέγοντας με τον κέρσορα κάποιον από τους νομούς, περνάει στο επόμενο επίπεδο το οποίο ονομάζεται Prefecture (Νομός). Σε αυτό το επίπεδο, εμφανίζεται ο νομός που έχει επιλεχτεί και διακριτές πινέζες(Pins) για κάθε μνημείο.Επιλέγοντας λοιπόν ο χρήστης κάποια από τις πινέζες, οδηγείται στο επόμενο επίπεδο το οποίο είναι το επίπεδο Region(οικισμός), και μπορεί να δεί το τρισδιάστατο μοντέλο του επιπέδου αυτού το οποίο περιέχει έναν αφαιρετικό σχεδιασμό του οικισμού του μνημείου με σκοπό να απεικονιστεί σε σχέση με όλη την οικιστική κτηριακή ομάδα που αποτελούσε την εκάστοτε πολιτισμική οντότητα. Κάθε οικισμός μπορεί να εμπεριέχει περισσότερα του ενός κτηριακά συγκροτήματα, τα οποία εντοπίζονται με την εμφάνιση πινέζας για το κάθε ένα από αυτά. Εκ νέου λοιπόν επιλέγοντας την εκάστοτε πινέζα από το επίπεδο-Regionοδηγούμαστε στο επίπεδο Complex(Κτηριακό συγκρότημα), το οποίο εμπεριέχει τα μνημεία τα οποία αποτελούν το κτηριακό συγκρότημα με μεγαλύτερη απόδοση λεπτομέρειας από το προηγούμενο επίπεδο, καθώς και κτήρια που δεν αποτελούσαν μνημεία αλλά είναι γνωστή η ύπαρξή τους στην περιοχή κοντά στα μνημεία. Κάθε μνημείο εντοπίζεται και επιλέγεται μέσω πινέζας, επιλέγοντας την οποία, περνάμε στο πέμπτο και τελευταίο επίπεδο της εφαρμογής, το οποίο είναι το επίπεδο Monument(Μνημείο). Σε αυτό το επίπεδο προβάλλεται στον χρήστη το μνημείο αυτό καθαυτό με επιπλέον λεπτομέρεια από τα προηγούμενα επίπεδα. Επιπλέον υπάρχει σε αυτό το επίπεδο η δυνατότητα ανάκτησης του χρήστη περεταίρω πληροφοριών για το μνημείο μέσω μίας αναδιπλούμενης καρτέλας η οποία δίνει στον χρήστη την δυνατότητα επιλογής μεταξύ των παρακάτω επιλογών:

i. Ιστορικά στοιχεία του μνημείου
ii. Βίντεο της περιοχής και του μνημείου
iii. Φωτογραφίες του μνημείου
iv. Χάρτης της περιοχής
v. Πληροφορίες επισκεψιμότητας
vi. Εξωτερικοί σύνδεσμοι (links) σε ιστοτόπους σχετικούς με το μνημείο.

Βασικό δομικό στοιχείο της εφαρμογής αποτελεί η μπάρα εναλλαγής μεταξύ επτά βασικών χρονικών περιόδων. Επιλέγοντας διαφορετικές περιόδους ο χρήστης μπορεί να συγκρίνει σε πραγματικό χρόνο τις μεταβολές που προκλήθηκαν στα μνημεία και τους οικισμούς μέσα στο πέρας του χρόνου, και να συγκρίνει την ανάπτυξη τους ανάλογα με την χρονική περίοδο. Οι βασικές αυτές ιστορικές περίοδοι είναι οι εξής:

i. Μινωική (Minoan)

ii. Ελληνιστική (Hellinistic)
iii. Ρωμαική (Roman)
iv. Βυζαντινή (Byzantine)
v. Ενετική (Venetian)
vi. Οθωμανική (Ottoman)
vii. Νεότερη (Modern)

Για την εφαρμογή αυτή όπως προείπαμε, απαιτούνταν η εισαγωγή 15 μνημείων σε όλα τα επίπεδα λεπτομέρειας και 70 μνημείων σημειακού χαρακτήρα(πινέζες). Η ιδέα μας και παράλληλα η πρόκληση για να έχει εξελικτικό χαρακτήρα η εφαρμογή ήταν η δημιουργία μιας επιπλέον εφαρμογής η οποία θα παρέχει στον χρήστη την δυνατότητα να εισάγει τα τρισδιάστατα μοντέλα που θέλει να προβάλει. Δημιουργήσαμε λοιπόν ένα διαδικτυακό εργαλείο για την διαχείριση της εφαρμογής και την επέκταση της μελλοντικά. Το εργαλείο αυτό είναι διαδικτυακό και επιτρέπει στον χρήστη να δημιουργήσει από την αρχή την διαδικτυακή εφαρμογή παρουσίασης των μνημείων μέσω φορμών που πρέπει να συμπληρώσει τα απαραίτητα στοιχεία και το ανέβασμα των τρισδιάστατων μοντέλων. Ανεβάζοντας κάποιο τρισδιάστατο μοντέλο, μέσω του εργαλείου διαχείρισης μπορεί να δεί το μοντέλο αυτό την ίδια στιγμή και να κάνει αλλαγές στην φόρμα με τα στοιχεία ώστε να ενημερωθεί στην βάση δεδομένων. Δημιουργήσαμε διαφορετικές καρτέλες για κάθε ιστορική εποχή, και ο χρήστης μεταβαίνοντας από καρτέλα σε καρτέλα μπορεί να αποφανθεί αν τον ικανοποιεί το αποτέλεσμα. Κάθε καρτέλα περιέχει μία φόρμα εισαγωγής στοιχείων για την κάθε ιστορική περίοδο. Στην φόρμα αυτή υπάρχει η επιλογή ονόματος, το οποίο εμφανίζεται αν θέλει ο διαχειριστής περνώντας τον κέρσορα πάνω από το τρισδιάστατο μοντέλο, η εισαγωγή τρισδιάστατου μοντέλου τύπου.obj συνοδευόμενο από το αρχείο .mtl αν υπάρχει υλικό στο μοντέλο, δυνατότητα επιλογής διαδραστικότητας με το μοντέλο, δηλαδή επιλέγοντας το αν συμβαίνει κάποια μετάβαση, εισαγωγή πατέρα του μοντέλου, δηλαδή μέσω ποιού μοντέλου βρέθηκε σε αυτό ο χρήστης, καθώς και επιλογή αλλαγής μεγέθους του, ώστε να εμφανίζεται με σωστές διαστάσεις στον τελικό χρήστη. Το εργαλείο αυτό επομένως παρέχει την δυνατότητα στον χρήστη να επεκτείνει τον αριθμό των μνημείων όλων των επιπέδων λεπτομέρειας, και να εισάγει νέα τρισδιάστατα μοντέλα. Αυτό το εργαλείο χρησιμοποιήθηκε από την εφαρμογή για να εισαχθούν όλα τρισδιάστατα μοντέλα και οι πληροφορίες στην βάση δεδομένων.

4.3 Χρήστες του συστήματος

Είναι πολύ δύσκολο να ορίσουμε συγκεκριμένες ομάδες χρηστών που αντιπροσωπεύουν αυτόν τον χρήστη, καθώς μια καθ' όλα «τρισδιάστατη» εφαρμογή δεν έχει συναντηθεί στο παρελθόν, και δεν μπορούμε να υποθέσουμε το αντίκτυπο που θα έχει στους επισκέπτες της. Θεωρήσαμε σκόπιμο βέβαια να προχωρήσουμε στην ανάλυση χρηστών και στις ομάδες οι οποίες τους αποτελούν, ώστε να μπορέσουμε να προσαρμόσουμε την ευχρηστία της εφαρμογής σε αυτές.

Έχουμε δύο είδη χρηστών:

- τον χρήστη-επισκέπτη της εφαρμογής
- τον χρήστη-διαχειριστή της εφαρμογής

Ο χρήστης-επισκέπτης, μπορεί να έχει πρόσβαση σε όλο το υλικό που παρέχεται από την εφαρμογή, τρισδιάστατα μοντέλα και επιμέρους πληροφορίες. Ουσιαστικά είναι ο χρήστης της βασικής εφαρμογής, που κάνει χρήση της για να γνωρίσει τα

μνημεία της Κρήτης. Η εφαρμογή αυτή μπορούμε να πούμε πως είναι τουριστικού αλλά και εκπαιδευτικού χαρακτήρα. Απευθύνεται κυρίως σε δύο μεγάλες ομάδες χρηστών, τους ταξιδιώτες-επισκέπτες οι οποίοι θέλουν να πάρουν πληροφορίες για τα ιστορικά μνημεία της Κρήτης πριν τα επισκεφτούν, και να δουν μια ευρύτερη εικόνα του μνημείου μέσω των οικισμών, καθώς και το που βρίσκονται, και πληροφορίες για την επισκεψιμότητα τους. Η δεύτερη κατηγορία είναι αυτή που καλύπτει τον εκπαιδευτικό χαρακτήρα της εφαρμογής και αφορά τους δασκάλους και εκπαιδευτικούς γενικότερα, οι οποίοι θέλουν μέσα από μια διαδραστική εφαρμογή να δείξουν στους μαθητές τους τον τρόπο με τον οποίο εξελίχτηκε κάποιο μνημείο, και φυσικά το ίδιο το μνημείο με τις επιμέρους πληροφορίες που παρέχονται.

Ο χρήστης διαχειριστής, στην προκειμένη έκδοση της εφαρμογής είχα μόνον εγώ αυτόν το ρόλο, είναι ο χρήστης εκείνος ο οποίος μέσα από το εργαλείο διαχείρισης που περιγράψαμε παραπάνω εισάγει τα τρισδιάστατα μοντέλα και τις πληροφορίες της εφαρμογής, καθώς και μπορεί να επεκτείνει τον όγκο των δεδομένων της εφαρμογής και τον προς παρουσίαση μνημείων.

4.4 Περιπτώσεις χρήσης (Use Cases)

Σε αυτή την ενότητα παρουσιάζεται η περιγραφή των περιπτώσεων χρήσης που υποστηρίζει το σύστημα. Οι περιπτώσεις χρήσεις είναι το σύνολο το σεναρίων που συνδέονται με ένα συγκεκριμένο σκοπό του χρήστη. Πρακτικά ορίζουν την διαδραστικότητα μεταξύ ενός χρήστη, και ενός συστήματος μέσω ακολουθίας βημάτων, για να επιτευχθεί ένας στόχος. Στόχος της περιγραφής των περιπτώσεων χρήσης είναι να περιγράψουν τις λειτουργικές απαιτήσεις του συστήματος και να δώσουν μία σαφή περιγραφή του τι θα πρέπει να κάνει το σύστημα ανάλογα με την αλληλεπίδραση του χρήστη. Οι περιπτώσεις χρήσης έχουν περιγραφεί στους παρακάτω πίνακες, με την μεθοδολογία του AlistairCockburn.

Use Case 1: “View Crete”	
Goal in Context	Ο χρήστης θέλει να επιλέξει na δεί το επίπεδο της Κρήτης.
Scope	Σύστημα
Actor	Χρήστης-Επισκέπτης
Preconditions	Ο χρήστης έχει βρίσκεται σε κάποιο άλλο επίπεδο λεπτομέρειας από το επίπεδο Crete.
Trigger	Ο χρήστης επιλέγει το κουμπί «Crete»
Description	1. Το σύστημα εμφανίζει το τρισδιάστατο μοντέλο του επιπέδου στο οποίο βρίσκεται. 2. Ο χρήστης επιλέγει το κουμπί «Crete» στον άξονα επιπέδου. 3. Το σύστημα ανακτά το τρισδιάστατο μοντέλο του επιλεγμένου επιπέδου και το εμφανίζει στον χρήστη.

Πίνακας 4.1 – Περίπτωση χρήσης 1, Προβολή επιπέδου Κρήτης

Use Case 2: “View Prefecture”

Goal in Context	Ο χρήστης θέλει να επιλέξει έναν νομό για περιήγηση.
Scope	Σύστημα
Actor	Χρήστης-Επισκέπτης
Preconditions	Ο χρήστης έχει επισκεφτεί την ιστοσελίδα της εφαρμογής.
Trigger	Ο χρήστης επιλέγει ένα νομό του τρισδιάστατου μοντέλου.
Description	1. Το σύστημα εμφανίζει την αρχική σελίδα με το επίπεδο λεπτομέρειας να είναι το Crete. 2. Ο χρήστης επιλέγει έναν από τους νομούς του τρισδιάστατου μοντέλου της Κρήτης. 3. Το σύστημα ανακτά το τρισδιάστατο μοντέλο του επιλεγμένου νομού και το εμφανίζει στον χρήστη.

Πίνακας 4.2 – Περίπτωση χρήσης 2, Επιλογή νομού

Use Case 3: "View Region"	
Goal in Context	Ο χρήστης θέλει να επιλέξει έναν οικισμό για να περιηγηθεί.
Scope	Σύστημα
Actor	Χρήστης-Επισκέπτης
Preconditions	Ο χρήστης έχει επιλέξει ήδη έναν νομό.
Trigger	Ο χρήστης επιλέγει μία πινέζα για περιήγηση στον οικισμό.
Description	1. Το σύστημα εμφανίζει τον νομό που είχε επιλεγεί σε προηγούμενο βήμα, και τις πινέζες των οικισμών πάνω σε αυτό. 2. Ο χρήστης επιλέγει μία πινέζα οικισμού. 3. Το σύστημα ανακτά το τρισδιάστατο μοντέλο του οικισμού και το εμφανίζει στον χρήστη.

Πίνακας 4.3 – Περίπτωση χρήσης 3, Επιλογή οικισμού

Use Case 3: "View Complex"	
Goal in Context	Ο χρήστης θέλει να επιλέξει ένα κτηριακό συγκρότημα για να περιηγηθεί σε αυτό.
Scope	Σύστημα
Actor	Χρήστης-Επισκέπτης
Preconditions	Ο χρήστης έχει επιλέξει ήδη έναν οικισμό σε προηγούμενο βήμα.

Trigger	Ο χρήστης επιλέγει μία πινέζα κτηριακού συγκροτήματος για να περιηγηθεί σε αυτό.
Description	1. Το σύστημα εμφανίζει τον οικισμό που έχει επιλεγεί σε προηγούμενο βήμα, και τις πινέζες των κτηριακών συγκροτημάτων του. 2. Ο χρήστης επιλέγει μία πινέζα κτηριακού συγκροτήματος. 3. Το σύστημα ανακτά το τρισδιάστατο μοντέλο του κτηριακού συγκροτήματος που έχει επιλεγεί, και το παρουσιάζει στον χρήστη.

Πίνακας 4.4 – Περίπτωση χρήσης 4, Επιλογή κτηριακού συγκροτήματος

Use Case 5: "View Monument"	
Goal in Context	Ο χρήστης θέλει να επιλέξει ένα μνημείο για να περιηγηθεί σε αυτό.
Scope	Σύστημα
Actor	Χρήστης- Επισκέπτης
Preconditions	Ο χρήστης έχει επιλέξει ήδη ένα κτηριακό συγκρότημα σε προηγούμενο βήμα.
Trigger	Ο χρήστης επιλέγει μία πινέζα μνημείου για να περιηγηθεί σε αυτό.
Description	1. Το σύστημα εμφανίζει το κτηριακό συγκρότημα με τις πινέζες μνημείων, που είχε επιλεγεί σε προηγούμενο βήμα. 2. Ο χρήστης επιλέγει μία πινέζα μνημείου. 3. Το σύστημα ανακτά το τρισδιάστατο μοντέλο του μνημείου και τις επιμέρους πληροφορίες του, και το παρουσιάζει στον χρήστη.

Πίνακας 4.5 – Περίπτωση χρήσης 5, Επιλογή μνημείου.

Use Case 6: "Change History Period"	
Goal in Context	Ο χρήστης θέλει να αλλάξει ιστορική περίοδο για να δεί την εξέλιξη ανά χρονολογική περίοδο.
Scope	Σύστημα
Actor	Χρήστης- Επισκέπτης
Preconditions	Ο χρήστης βρίσκεται σε ένα από τα επίπεδα: • Νομός- (Prefecture) • Οικισμός- (Region) • Κτηριακό συγκροτήματα- (Complex) • Μνημείο – (Monument)
Trigger	Ο χρήστης επιλέγει μία από της εποχές:

	• Μινωική (Minoan) • Ελληνιστική (Hellinistic) • Ρωμαική (Roman) • Βυζαντινή (Byzantine) • Ενετική (Venetian) • Οθωμανική (Ottoman) • Νεότερη (Modern)
Description	1. Το σύστημα εμφανίζει το επίπεδο που έχει προεπιλεγεί από τον χρήστη, στην χρονική περίοδο που είχε προεπιλεγεί. 2. Ο χρήστης επιλέγει μία άλλη περίοδο από την ήδη επιλεγμένη. 3. Το σύστημα ανακτά το επιλεγμένο τρισδιάστατο μοντέλο του επιπέδου στην διαφορετική χρονική περίοδο, και το παρουσιάζει στον χρήστη.

Πίνακας 4.6 – Περίπτωση χρήσης 6, επιλογή εναλλαγήςχρονικής περιόδου

Use Case 7: "View Hidden Menu"	
Goal in Context	Ο χρήστης θέλει να δεί το μενού με τις επιμέρους πληροφορίες του μνημείου.
Scope	Σύστημα
Actor	Χρήστης- Επισκέπτης
Preconditions	Ο χρήστης έχει επιλέξει ένα μνημείο στο επίπεδο λεπτομέρειας Monument.
Trigger	Ο χρήστης επιλέγει το «βέλος» ανοίγματος του μενού.
Description	1. Το σύστημα εμφανίζει το μνημείο που έχει επιλεγεί σε προηγούμενο βήμα σε οποιαδήποτε εποχή. 2. Ο χρήστης επιλέγει το βέλος στα δεξιά για να ανοίξει το κρυφό μενού πληροφοριών. 3. Το σύστημα ανταποκρίνεται στην ενέργεια του χρήστη και ανοίγει η καρτέλα-μενού με τις επιλογές: i. Ιστορικά στοιχεία του μνημείου ii. Βίντεο της περιοχής και του μνημείου iii. Φωτογραφίες του μνημείου iv. Χάρτης της περιοχής v. Πληροφορίες επισκεψιμότητας vi. Εξωτερικοί σύνδεσμοι (links) σε ιστοτόπους

σχετικούς με το μνημείο.

Πίνακας 4.7 -Περίπτωση χρήσης 7, επιλογή προβολής του κρυφού μενού

Use Case 8: "View History Information"	
Goal in Context	Ο χρήστης θέλει να δεί τις ιστορικές πληροφορίες που παρέχονται για το μνημείο που έχει επιλέξει.
Scope	Σύστημα
Actor	Χρήστης- Επισκέπτης
Preconditions	Ο χρήστης έχει «ανοίξει» το κρυφό μενού σε προηγούμενο βήμα.
Trigger	Ο χρήστης επιλέγει το κουμπί: History
Description	1. Το σύστημα εμφανίζει το μνημείο στην οθόνη και στο δεξί μέρος έχει ανοίξει το κρυφό μενού. 2. Ο χρήστης επιλέγει το κουμπί «History» για να προβληθούν οι ιστορικές πληροφορίες που αφορούν το μνημείο. 3. Το σύστημα ανακτά τις ιστορικές πληροφορίες του μνημείου και ανοίγει επιμέρους καρτέλα στην οποία εμφανίζονται οι πληροφορίες αυτές.

Πίνακας 4.8-Περίπτωση χρήσης 8, επιλογή προβολής Ιστορικών πληροφοριών σχετικών με το μνημείο.

Use Case 9: "View Video"	
Goal in Context	Ο χρήστης θέλει να δεί βίντεο σχετικό με το μνημείο.
Scope	Σύστημα
Actor	Χρήστης- Επισκέπτης
Preconditions	Ο χρήστης έχει «ανοίξει» το κρυφό μενού σε προηγούμενο βήμα.
Trigger	Ο χρήστης επιλέγει το κουμπί: Video
Description	1. Το σύστημα εμφανίζει το μνημείο στην οθόνη και στο δεξί μέρος έχει ανοίξει το κρυφό μενού. 2. Ο χρήστης επιλέγει το κουμπί «Video» για να προβληθεί το σχετικό βίντεο. 3. Το σύστημα ανακτά το σχετικό βίντεο του μνημείου και ανοίγει επιμέρους καρτέλα στην οποία παρουσιάζεται.

Πίνακας 4.9-Περίπτωση χρήσης 9, επιλογή προβολής βίντεο σχετικό με το μνημείο.

Use Case 10: "View Photos"	
Goal in Context	Ο χρήστης θέλει να δεί τις σχετικές φωτογραφίες που

	παρέχονται για το μνημείο που έχει επιλέξει.
Scope	Σύστημα
Actor	Χρήστης- Επισκέπτης
Preconditions	Ο χρήστης έχει «ανοίξει» το κρυφό μενού σε προηγούμενο βήμα.
Trigger	Ο χρήστης επιλέγει το κουμπί: Photos
Description	1. Το σύστημα εμφανίζει το μνημείο στην οθόνη και στο δεξί μέρος έχει ανοίξει το κρυφό μενού. 2. Ο χρήστης επιλέγει το κουμπί «Photos» για να προβληθούν φωτογραφίες που αφορούν το μνημείο. 3. Το σύστημα ανακτά τις φωτογραφίες του μνημείου και ανοίγει επιμέρους καρτέλα στην οποία εμφανίζονται οι φωτογραφίες αυτές.

Πίνακας 4.10-Περίπτωση χρήσης 10, επιλογή προβολής φωτογραφιών σχετικών με το μνημείο.

Use Case 11: "View Map"	
Goal in Context	Ο χρήστης θέλει να δεί την τοποθεσία του μνημείου σε πραγματικό χάρτη.
Scope	Σύστημα
Actor	Χρήστης- Επισκέπτης
Preconditions	Ο χρήστης έχει «ανοίξει» το κρυφό μενού σε προηγούμενο βήμα.
Trigger	Ο χρήστης επιλέγει το κουμπί: Map
Description	1. Το σύστημα εμφανίζει το μνημείο στην οθόνη και στο δεξί μέρος έχει ανοίξει το κρυφό μενού. 2. Ο χρήστης επιλέγει το κουμπί «Map» για να εμφανιστεί ο χάρτης με την τοποθεσία του συγκεκριμένου μνημείου. 3. Το σύστημα ανακτά τον χάρτη του μνημείου και ανοίγει επιμέρους καρτέλα στην οποία εμφανίζεται ο χάρτης του μνημείου.

Πίνακας 4.11-Περίπτωση χρήσης 11, επιλογή προβολής χάρτη του μνημείου

Use Case 12: "View Information"	
Goal in Context	Ο χρήστης θέλει να δεί συγκεκριμένες πληροφορίες σχετικά με την επισκεψιμότητα του μνημείου.
Scope	Σύστημα
Actor	Χρήστης – Επισκέπτης
Preconditions	Ο χρήστης έχει «ανοίξει» το κρυφό μενού σε προηγούμενο βήμα.

Trigger	Ο χρήστης επιλέγει το κουμπί: «Information»
Description	1. Το σύστημα εμφανίζει το μνημείο στην οθόνη και στο δεξί μέρος έχει ανοίξει το κρυφό μενού. 2. Ο χρήστης επιλέγει το κουμπί «Information» για να εμφανιστούν γενικές πληροφορίες και πληροφορίες που αφορούν την επισκεψιμότητα του μνημείου. 3. Το σύστημα ανακτά τις πληροφορίες αυτές και ανοίγει μία επιμέρους καρτέλα όπου εμφανίζονται οι επιμέρους αυτές πληροφορίες.

Πίνακας 4.12-Περίπτωση χρήσης 12, επιλογή προβολής γενικών πληροφοριών σχετικών με το μνημείο και την επισκεψιμότητα του.

Use Case 13: "View External links"	
Goal in Context	Ο χρήστης θέλει να δεί εξωτερικούς συνδέσμους σχετικά με το μνημείο.
Scope	Σύστημα
Actor	Χρήστης – Επισκέπτης
Preconditions	Ο χρήστης έχει «ανοίξει» το κρυφό μενού σε προηγούμενο βήμα.
Trigger	Ο χρήστης επιλέγει το κουμπί: «Externallinks»
Description	1. Το σύστημα εμφανίζει το μνημείο στην οθόνη και στο δεξί μέρος έχει ανοίξει το κρυφό μενού. 2. Ο χρήστης επιλέγει το κουμπί «Externallinks» για να εμφανιστούν εξωτερικοί σύνδεσμοι σχετικοί με το μνημείο. 3. Το σύστημα ανακτά τους εξωτερικούς συνδέσμους και ανοίγει επιμέρους καρτέλα η οποία τους παρουσιάζει.

Πίνακας 4.13-Περίπτωση χρήσης 13, επιλογή προβολής εξωτερικών συνδέσμων σχετικών με το μνημείο.

Οι παραπάνω περιπτώσεις χρήσης (πίνακες 4.1 έως 4.12), αφορούν την βασική εφαρμογή της διπλωματικής αυτής, που περιλαμβάνει την παρουσίαση των μνημείων ώς τρισδιάστατων μοντέλων στον φυλλομετρητή του χρήστη-επισκέπτη. Παρακάτω θα παρουσιάσουμε τις περιπτώσεις χρήσης της εφαρμογής διαχείρισης, η οποία όπως θα φανεί και στους πίνακες αφορά τον χρήστη-διαχειριστή.

Use Case 14: "Insert Ground"	
Goal in Context	Ο χρήστης θέλει να εισάγει το έδαφος του τρισδιάστατου μοντέλου.
Scope	Σύστημα
Actor	Χρήστης – Διαχειριστής
Preconditions	Ο χρήστης έχει επισκεφτεί την ιστοσελίδα διαχείρισης.

Trigger	Ο χρήστης επιλέγει την καρτέλα «ground».
Description	1. Το σύστημα παρουσιάζει στον χρήστη 8 καρτέλες με επιλεγμένη την καρτέλα ground. Η καρτέλα περιέχει μια φόρμα εισαγωγής στοιχείων. 2. Ο χρήστης συμπληρώνει κατάλληλα την φόρμα. 3. Ο χρήστης πατάει το κουμπί “InsertGround” 4. Το σύστημα δημιουργεί το αντικείμενο εδάφους με τα στοιχεία του, τα μεταφέρει στην βάση δεδομένων και τα επανακτά από αυτήν για την εμφάνιση του τρισδιάστατου μοντέλου σε previewmode.

Πίνακας 4.14 – Περίπτωση χρήσης 14, εισαγωγή εδάφους τρισδιάστατου μοντέλου.

Use Case 15: “Add Pin”	
Goal in Context	Ο χρήστης θέλει να προσθέσει μία πινέζα, η οποία συνδέεται με το έδαφος.
Scope	Σύστημα
Actor	Χρήστης – Διαχειριστής
Preconditions	Ο χρήστης έχει εισάγει τρισδιάστατο μοντέλο εδάφους σε προηγούμενο βήμα.
Trigger	Ο χρήστης επιλέγει το κουμπί «AddPin».
Description	1. Το σύστημα παρουσιάζει στον χρήστη μία φόρμα ώστε να δημιουργηθεί ένα νέο αντικείμενο πινέζας. 2. Ο χρήστης συμπληρώνει τα απαραίτητα πεδία της φόρμας και επιλέγει το κουμπί InsertPin. 3. Το Σύστημα δημιουργεί το αντικείμενο πινέζας, το αποθηκεύει στην βάση δεδομένων. 4. Το σύστημα ανακτά την αποθηκευμένη πινέζα και την εμφανίζει στο προ υπάρχονpreviewmodeμαζί με τα άλλα τρισδιάστατα μοντέλα.

Πίνακας 4.15 – Περίπτωση χρήσης 15, προσθήκη πινέζας στην υπάρχουσα σκηνή.

Use Case 16: “Insert Period Model”	
Goal in Context	Ο χρήστης θέλει να εισάγει το τρισδιάστατο μοντέλο που αφορά μία από τις εποχές.
Scope	Σύστημα
Actor	Χρήστης – Διαχειριστής
Preconditions	Ο χρήστης έχει εισάγει τρισδιάστατο μοντέλο εδάφους σε προηγούμενο βήμα.
Trigger	Ο χρήστης επιλέγει μία από τις καρτέλες:

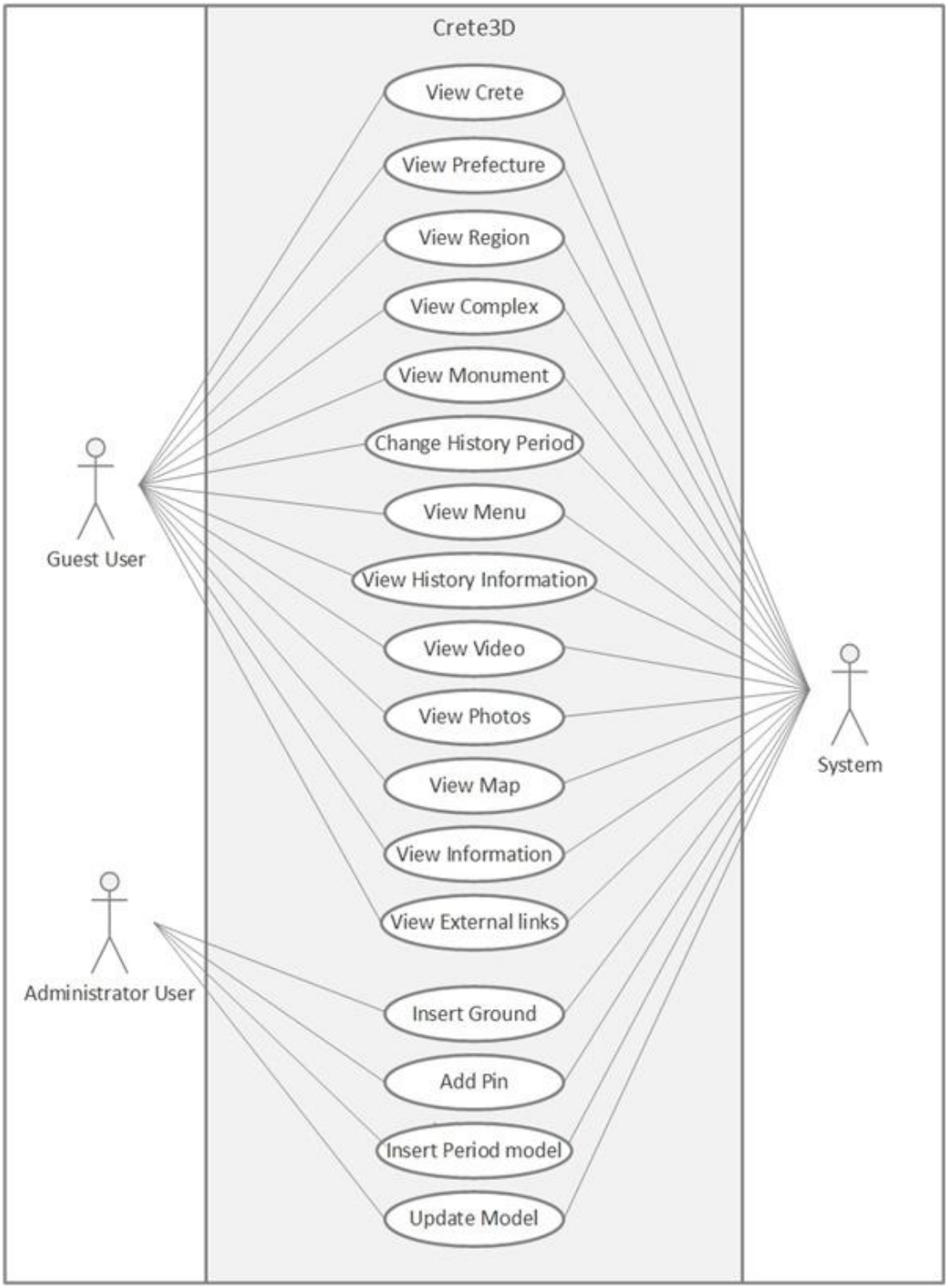
	i. Μινωική (Minoan) ii. Ελληνιστική (Hellinistic) iii. Ρωμαική (Roman) iv. Βυζαντινή (Byzantine) v. Ενετική (Venetian) vi. Οθωμανική (Ottoman) vii. Νεότερη (Modern)
Description	1. Το σύστημα παρουσιάζει στον χρήστη την φόρμα η οποία αφορά την εισαγωγή τρισδιάστατου μοντέλου σε σχέση με την επιλογή της εποχής. 2. Ο χρήστης συμπληρώνει κατάλληλα την φόρμα. 3. Ο χρήστης πατάει το κουμπί “InsertModel”. 4. Το σύστημα δημιουργεί το αντικείμενο του μοντέλου, μεταφέρει τα δεδομένα στην βάση δεδομένων και τα επανακτά από αυτήν για την εμφάνιση του τρισδιάστατου μοντέλου σε previewmode σε συνδυασμό με τα υπόλοιπα τρισδιάστατα μοντέλα της σκηνής.

Πίνακας 4.16 – Περίπτωση χρήσης 16, εισαγωγή τρισδιάστατου μοντέλου ανάλογα με την εποχή

Use Case 17: “Update Model”	
Goal in Context	Ο χρήστης θέλει να ανανεώσει το τρισδιάστατο μοντέλο που έχει εισάγει.
Scope	Σύστημα
Actor	Χρήστης – Διαχειριστής
Preconditions	Ο χρήστης έχει εισάγει τρισδιάστατο μοντέλο εδάφους ή περιόδου σε προηγούμενο βήμα.
Trigger	Ο χρήστης επιλέγει το κουμπί «UpdateModel».
Description	1. Το σύστημα παρουσιάζει στον χρήστη την φόρμα η οποία περιέχει τα στοιχεία του τρισδιάστατου μοντέλου που έχει επιλέξει, καθώς και το previewmodeπαράθυρο. 2. Ο χρήστης κάνει αλλαγές στην φόρμα. 3. Ο χρήστης πατάει το κουμπί “UpdateModel”. 4. Το σύστημα εντοπίζει τις αλλαγές στο αντικείμενο του μοντέλου, και αντικαθιστά τα δεδομένα στην βάση δεδομένων και τα επανακτά από αυτήν για την εμφάνιση του τρισδιάστατου μοντέλου σε previewmode σε συνδυασμό με τα υπόλοιπα τρισδιάστατα μοντέλα της σκηνής, έχοντας τις αλλαγές του χρήστη.

Πίνακας 4.17 – Περίπτωση χρήσης 17, ενημέρωση δεδομένων αντικειμένου της σκηνής.

Στην παρακάτω εικόνα(εικόνα 4.1) παρουσιάζεται το διάγραμμα UML που περιγράφει συνολικά τις περιπτώσεις χρήσεις των χρηστών.



Εικόνα 4.1- UML διάγραμμά περιπτώσεων χρήσης.

5 Υλοποίηση

5.1 Εισαγωγή

Στα προηγούμενα κεφάλαια παρουσιάσαμε αναλυτικά τις τεχνολογίες που χρησιμοποιήσαμε για την υλοποίηση της εφαρμογής και τις λειτουργικές της απαιτήσεις. Στο κεφάλαιο αυτό θα αναλύσουμε τον τρόπο υλοποίησης της εφαρμογής μέσω των τεχνολογιών και θα παραθέσουμε σημαντικά κομμάτια κώδικα επεξηγώντας την χρήση τους για την υλοποίηση. Πιο συγκεκριμένα στο κεφάλαιο αυτό θα αναλύσουμε την βάση δεδομένων, τον κώδικα που ήταν απαραίτητος για την λειτουργικότητα της εφαρμογής, τον τρόπο επικοινωνίας του διακομιστή με τον χρήστη-πελάτη, και τον κώδικα υλοποίησης της γραφικής διεπαφής της εφαρμογής, και το πιο σημαντικό σε αυτή την διπλωματική την υλοποίηση της τρισδιάστατης σκηνής στον φυλλομετρητή μέσω της WebGL.

5.2 Βάση Δεδομένων

Η βάση δεδομένων αποτελεί βασικό πυλώνα της εφαρμογής, καθώς στόχος μας ήταν η επεκτασιμότητα της εφαρμογής και όχι ένα στατικό αποτέλεσμα. Αυτό θα μπορούσε να επιτευχθεί μόνο με την χρήση βάσης δεδομένων η οποία θα συνδέει όλα τα δεδομένα των τρισδιάστατων μοντέλων και τις επιμέρους πληροφορίες που παρέχονται. Χρησιμοποιήσαμε για την υλοποίηση σχεσιακή βάση δεδομένων και συγκεκριμένα την MySQL.

5.2.1 Σχεδιασμός βάσης δεδομένων

Σημαντικό βήμα για την κατασκευή βάσης δεδομένωνείναι ο σχεδιασμός της και ο διαχωρισμός των οντοτήτων οι οποίες την αποτελούν. Για την περιγραφή της βάσης δεδομένων που υλοποιήσαμε, χρησιμοποιήσαμε το διάγραμμα Entity-Relationship(E-R) το οποίο είναι ένα αφαιρετικό διάγραμμα κατανόησης της βάσης δεδομένων που χρησιμοποιήθηκε στην εφαρμογή μας. Στο διάγραμμα που υλοποιήσαμε (εικόνα 5.1) χρησιμοποιούνται οι εξής έννοιες:

- Οντότητα (Entity): Είναι ένα αντικείμενο στον πραγματικό κόσμο το οποίο ξεχωρίζει από τα υπόλοιπα. Οντότητες μπορεί να είναι αντικείμενα, άνθρωποι, τοποθεσίες και άλλα. Στο διάγραμμα E-R αναπαριστώνται με ένα ορθογώνιο.
- Χαρακτηριστικό (attribute): Κάθε οντότητα έχει διάφορα στοιχεία που την προσδιορίζουν. Η οντότητα παριστάνεται στο διάγραμμα ως έλλειψη. Υπογραμμισμένο χαρακτηριστικό μέσα στην έλλειψη είναι το πρωτεύον κλειδί, το οποίο είναι μοναδικό για κάθε στιγμιότυπο της.
- Συσχέτιση (relationship): Συσχέτιση είναι η σύνδεση μεταξύ δύο οντοτήτων και παριστάνεται ώς ρόμβος.
- Πληθικότητα (Cardinality): Η πληθικότητα περιγράφει τον αριθμό στιγμιότυπωνμίας οντότητας που μπορεί να αντιστοιχίζεται με μία άλλη σε μία συσχέτιση. Μπορεί να έχουμε συσχετίσεις με λόγο πληθικότητας:
    i. 1-1 (ένα προς ένα):

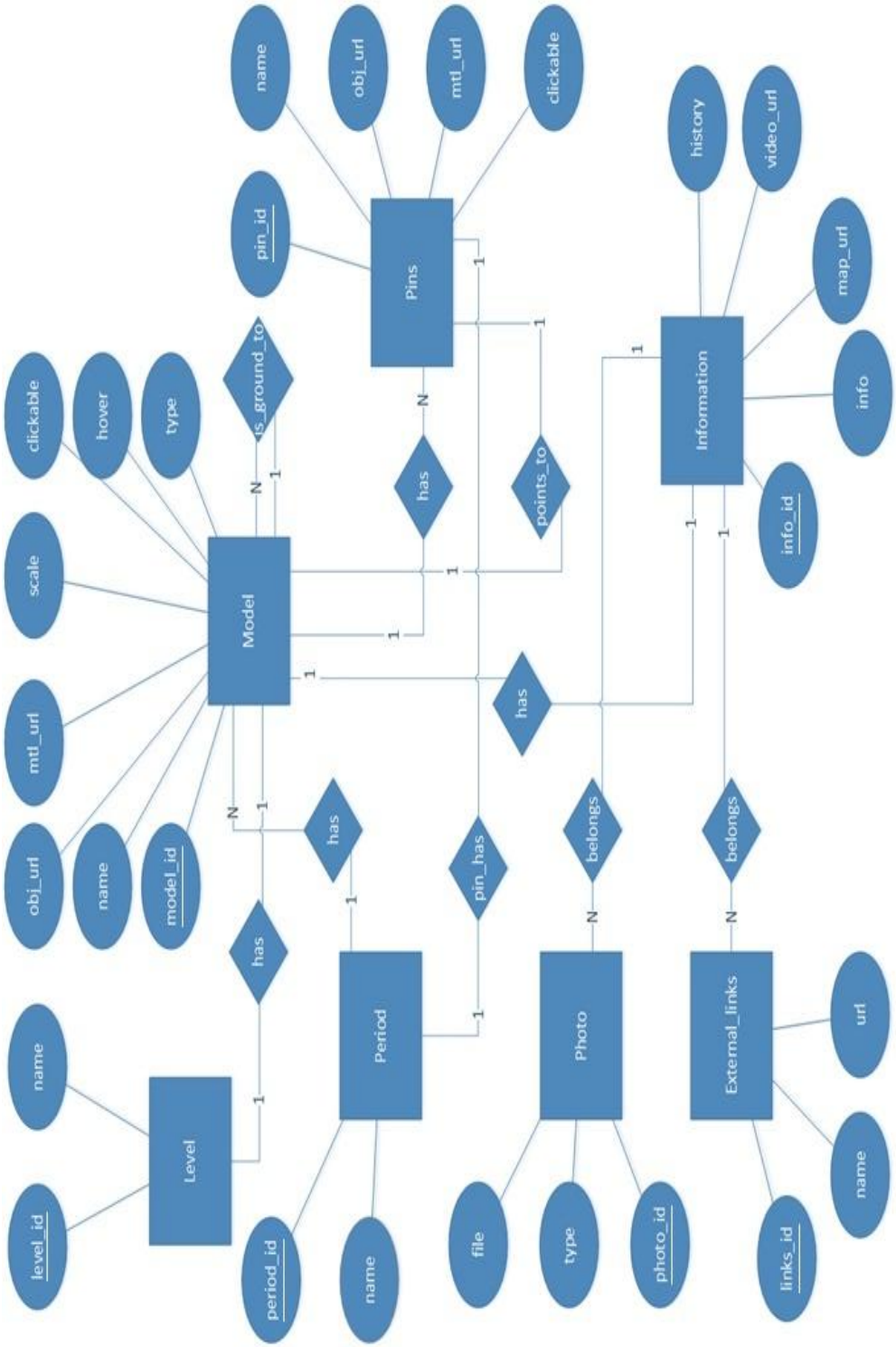
       Αντιστοιχίζεται μία οντότητα ενός τύπου με το πολύ μία οντότητα ενός άλλου τύπου.

    ii. 1-N (ένα προς πολλά):

Αντιστοιχίζεται μία οντότητα ενός τύπου με κανένα, ένα ή πολλά στιγμιότυπα οντότητας άλλου τύπου.

iii. M-N (πολλά προς πολλά):

Αντιστοιχίζεται κάθε στιγμιότυπο οντότητας ενός τύπου με κανένα, ένα ή πολλά στιγμιότυπα άλλου τύπου.



Εικόνα 5.1 -Διάγραμμα Entity-Relationshiptης βάσης δεδομένων

Το διάγραμμα Entity-Relationship, με την χρήση συγκεκριμένων κανόνων μετατρέπεται σε RelationalSchema, το οποίο είναι μία δομή οργάνωσης της βάσης δεδομένων και εξηγεί πως είναι δομημένη η βάση, χωρίζοντας τους πίνακες που απαιτούνται για την υλοποίηση της. Οι κανόνες μετατροπής του διαγράμματος E-Rσε RelationalSchemaπου χρησιμοποιήθηκαν είναι οι εξής:

i. Για κάθε απλή οντότητα E του ERδημιουργούμε μία σχέση Rπου περιλαμβάνει όλες τις απλές ιδιότητες του E και θέτουμε το πρωτεύον κλειδί της R.

ii. Για κάθε 1-1 σχέση Rτου ER, βρίσκουμε τις σχέσεις βρίσκουμε τις σχέσεις Sκαι Tπου αντιστοιχούν στις οντότητες που συμμετέχουν στην Rκαι προσθέτουμε σε μία από τις δύο σαν ξένο κλειδί (ForeignKey) το πρωτεύον κλειδί της άλλης.

iii. Για κάθε N-1 σχέση Rτου ERβρίσκουμε τη σχέση Sπου συμμετέχει στη Νπλευρά της Rκαι της προσθέτουμε σαν foreignkeyτο πρωτεύον κλειδί της σχέσης T που βρίσκεται από την άλλη πλευρά.

iv. Για κάθε N-M σχέση Rδημιουργούμε μία οντότητα Sγια να αναπαραστήσουμε την Rκαι βάζουμε σαν foreignkeysστην Sτα πρωτεύον κλειδιά των οντοτήτων που συμμετέχουν στην σχέση. Ο συνδυασμός αυτός αποτελεί και το πρωτεύον κλειδί της S.

Με βάση τους κανόνες αυτούς το RelationalSchemaμας αποτυπώνεται ως εξής:

Model(<u>model_id</u>, name, obj_url, mtl_url, clickable, hover, type,scale,period_id^{FK}, level_id^{FK}, ground_id^{FK})

Pins(<u>pin_id</u>, name, obj_url,mtl_url, clickable,period_id^{FK}, ground_id^{FK},points_id^{FK})

Level(<u>level_id</u>,name)

Period(<u>period_id</u>, name)

Information(<u>info_id</u>, info, map_url,video_url,history,model_id^{FK})

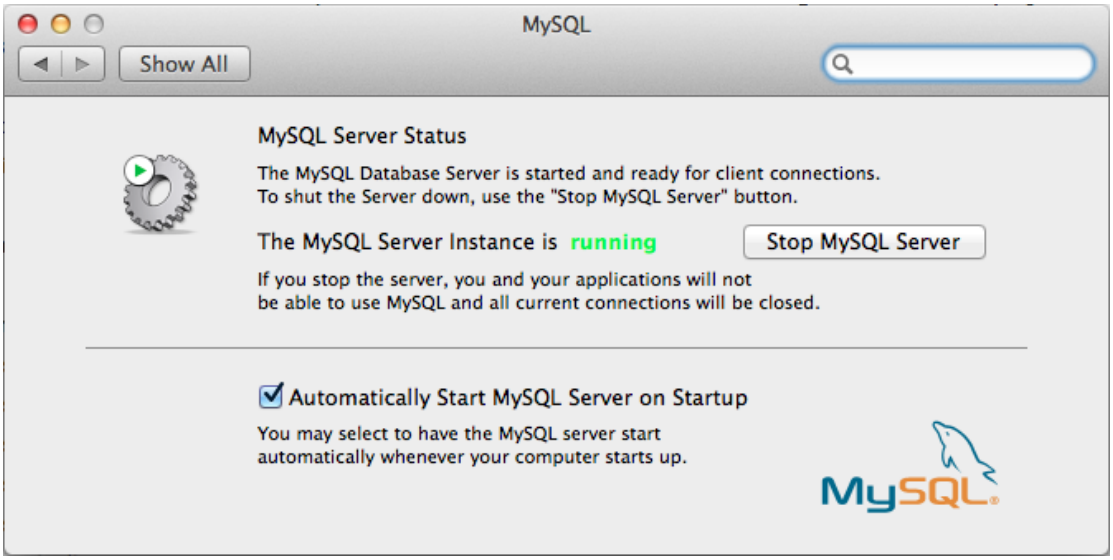
Photo(<u>photo_id</u>,file,type,info_id^{FK})

External_links(<u>links_id</u>, name, url, info_id^{FK})

Στο RelationalSchema, αποτυπώσαμε ως ground_id και points_idτο πρωτεύον κλειδίmodel_id της οντότητας Model, για να μην υπάρχει σύγχυση με τα ίδια ονόματα.Υπογραμμισμένα είναι τα πρωτεύον κλειδιά, και με δείκτη (^{FK}) τα ξένα κλειδιά (foreignkeys). Από το RelationalSchema, προέκυψαν οι πίνακες τους οποίους έπρεπε να κατασκευάσουμε για την βάση δεδομένων που χρειαζόμαστε. Αυτοίείναιόπωςφαίνεταιοι : Model, Pins, Level, Period, Information, Photo, External_Links.

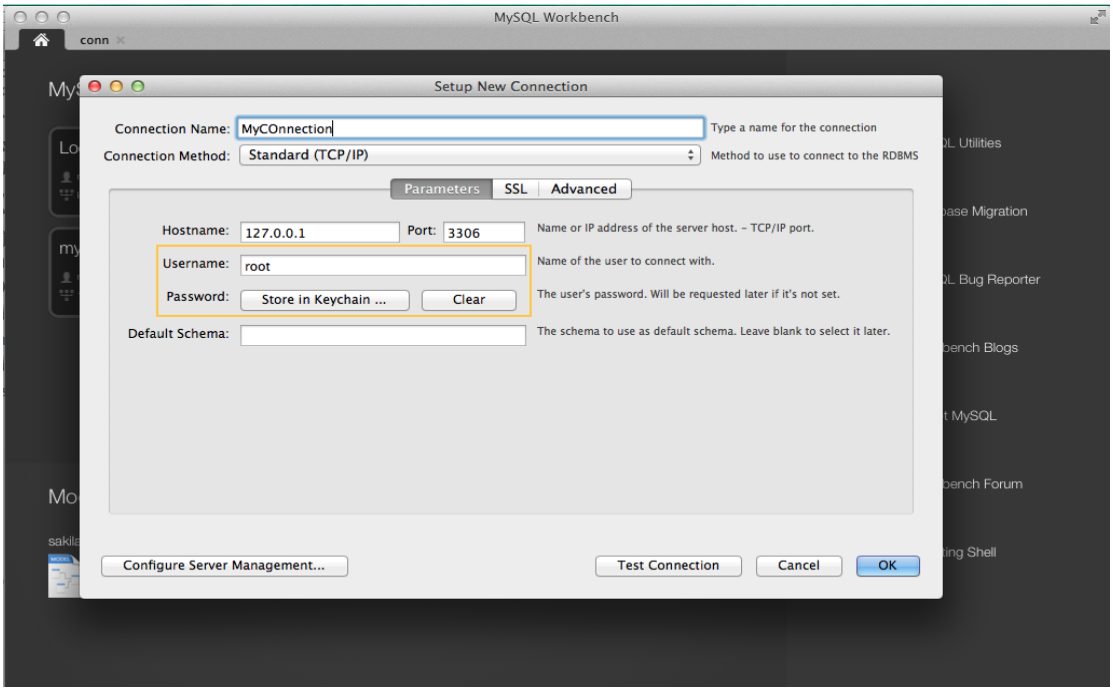
5.2.2 MySQL

Σε αυτή την ενότητα θα εξηγήσουμε πως από τον σχεδιασμό της βάσης δεδομένων της προηγούμενης ενότητας, υλοποιήσαμε την βάση δεδομένων μέσω του συστήματος διαχείρισης MySQL. Το πρώτο βήμα που πραγματοποιήσαμε για την κατασκευή της βάσης δεδομένων ήταν η εγκατάσταση του Mysqlservero οποίος φιλοξενεί την βάση δεδομένων που θα υλοποιήσουμε στα επόμενα βήματα. Εγκαθιστώντας τον MySQLServer μπορούμε να τον ξεκινήσουμε και να τον σταματήσουμε μέσω της κονσόλας ή μέσω του παραθύρου που φαίνεται παρακάτω (εικόνα 5.2).



Εικόνα5.2 - MySQL server status (κατάστασηMySQL server)

Στην συνέχεια εγκαταστήσαμε το εργαλείο γραφικής διαχείρισης της βάσης δεδομένων MySQLWorkbench στο οποίο δημιουργήσαμε μία σύνδεση με τον MySQLserver (εικόνα 5.3) θέτοντας τις απαραίτητες ρυθμίσεις και ορίζοντας το όνομα χρήστη και τον κωδικό του διαχειριστή της σύνδεσης.



Εικόνα 5.3 – MySQLworkbench, δημιουργία σύνδεσης.

Έχοντας ακολουθήσει τα παραπάνω βήματα, είμαστε πλέον έτοιμοι να δημιουργήσουμε την βάση δεδομένων μας, σύμφωνα με το σχεσιακό σχήμα (RelationalSchema), γράφοντας τον απαραίτητο SQL κώδικα.

```
1 DROP SCHEMA IF EXISTS `crete3Ddb`;
2 CREATE SCHEMA IF NOT EXISTS `crete3Ddb` DEFAULT CHARACTER SET utf8;
3 USE `crete3Ddb`;
```

Εικόνα 5.4 – Κώδικας της βάσης δεδομένων (μέρος πρώτο).

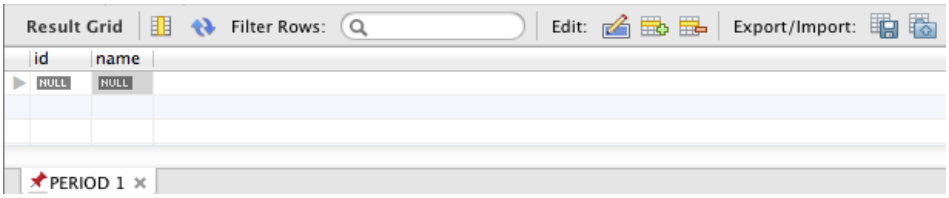
Στον παραπάνω SQLκώδικα (εικόνα 5.4), δημιουργούμε το «σχήμα» της βάσης δεδομένων μας, αφού πρώτα έχουμε διαγράψει τυχόν προηγούμενο με το ίδιο όνομα, και το θέτουμε ώς το βασικό σχήμα στο οποίο θα τρέχουν τα SQLscripts.Στην συνέχεια, γράψαμε τον κώδικα για την δημιουργία καθενός από τους πίνακες που προέκυψαν από το Relational-Schema, ορίζοντας το πρωτεύον και ξένο κλειδί του καθενός, καθώς και τον τύπο του κάθε στοιχείου.

Πίνακας Period:

```
1 -- -----------------------------------------------------
2 -- Table `crete3Ddb`.`Period`
3 -- -----------------------------------------------------
4 CREATE  TABLE IF NOT EXISTS `crete3Ddb`.`PERIOD` (
5   `id` INT(11) NOT NULL AUTO_INCREMENT,
6   `name` VARCHAR(45) NOT NULL,
7   PRIMARY KEY (`id`)
8   );
```

Εικόνα 5.5 - MySQLδημιουργία πίνακα Period

Αποτέλεσμα στο γραφικό εργαλείο σχεδιασμού MySQLWorkbench:



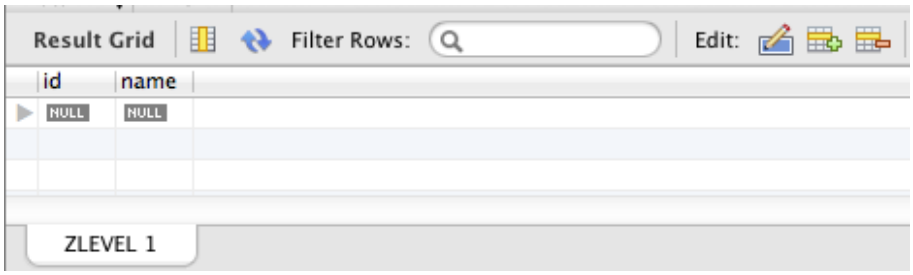
Εικόνα 5.6 – Πίνακας Periodτης βάσης δεδομένων.

Πίνακας Level:

```
-- -----------------------------------------------------
-- Table `crete3Ddb`.`ZLevel`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `crete3Ddb`.`ZLEVEL` (
  `id` INT(11) NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`id`)
  );
```

Εικόνα 5.7 – MySQLδημιουργία πίνακα Level

Αποτέλεσμα στο γραφικό εργαλείο σχεδιασμού MySQLWorkbench:



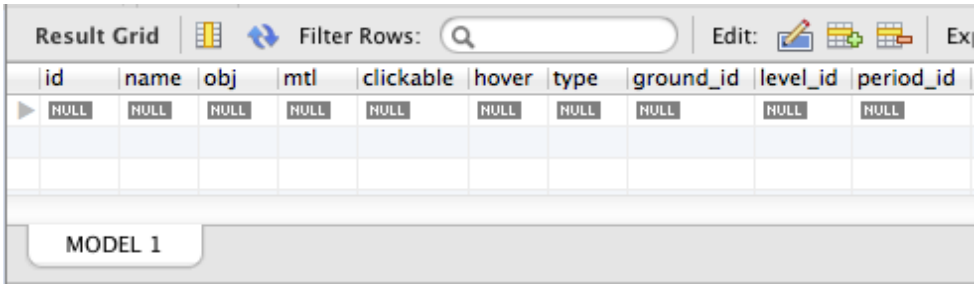
Εικόνα 5.8 - Πίνακας Levelτης βάσης δεδομένων

Πίνακας Model:

```
-- -----------------------------------------------------
-- Table `crete3Ddb`.`MODEL`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `crete3Ddb`.`MODEL` (
  `id` INT(11) NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(100) NOT NULL UNIQUE,
  `obj`  VARCHAR(200) NOT NULL,
  `mtl`  VARCHAR(200) NOT NULL,
  `clickable` ENUM('TRUE','FALSE'),
  `hover` ENUM('TRUE','FALSE'),
  `type` ENUM('MODEL','GROUND'),
  `ground_id` INT(11),
  `level_id` INT(11),
  `period_id` INT(11),
  PRIMARY KEY (`id`),
   CONSTRAINT `GROUND_FK` FOREIGN KEY (`ground_id`)
   REFERENCES MODEL(id) ON DELETE NO ACTION ON UPDATE CASCADE,
   CONSTRAINT `PERIOD_FK` FOREIGN KEY (`period_id`)
    REFERENCES PERIOD(id) ON DELETE NO ACTION ON UPDATE CASCADE,
   CONSTRAINT `LEVEL_FK` FOREIGN KEY (`level_id`)
   REFERENCES ZLEVEL(id) ON DELETE NO ACTION ON UPDATE CASCADE
  );
```

Εικόνα 5.9 - MySQLδημιουργία πίνακα Model

Αποτέλεσμα στο γραφικό εργαλείο σχεδιασμού MySQLWorkbench:



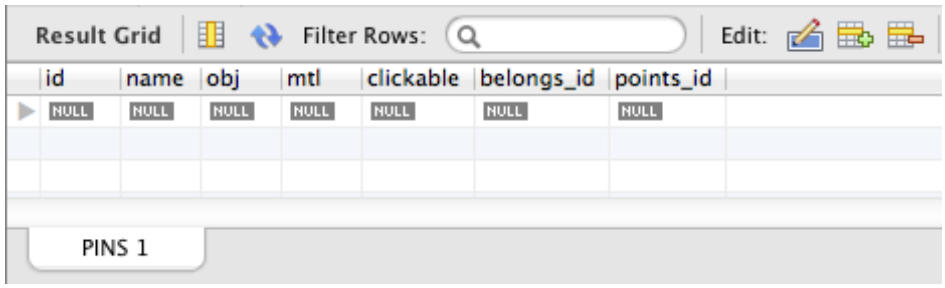
Εικόνα 5.10 - Πίνακας Modelτης βάσης δεδομένων

Πίνακας Pins:

```
-- -----------------------------------------------------
-- Table `crete3Ddb`.`Pins`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `crete3Ddb`.`PINS` (
  `id` INT(11) NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(100) NOT NULL UNIQUE,
  `obj`  VARCHAR(200) ,
  `mtl`  VARCHAR(200) ,
  `clickable` ENUM('TRUE','FALSE'),
  `belongs_id` INT(11),
  `points_id` INT(11),
  PRIMARY KEY (`id`),
  CONSTRAINT `MODEL_BELONGS_FK` FOREIGN KEY (`belongs_id`)
  REFERENCES MODEL(id) ON DELETE NO ACTION ON UPDATE CASCADE,
  CONSTRAINT `MODEL_POINTS_TO_ID` FOREIGN KEY (`points_id`)
  REFERENCES MODEL(id) ON DELETE NO ACTION ON UPDATE CASCADE
  );
```

Εικόνα 5.11 - MySQLδημιουργία πίνακα Pins

Αποτέλεσμα στο γραφικό εργαλείο σχεδιασμού MySQLWorkbench:



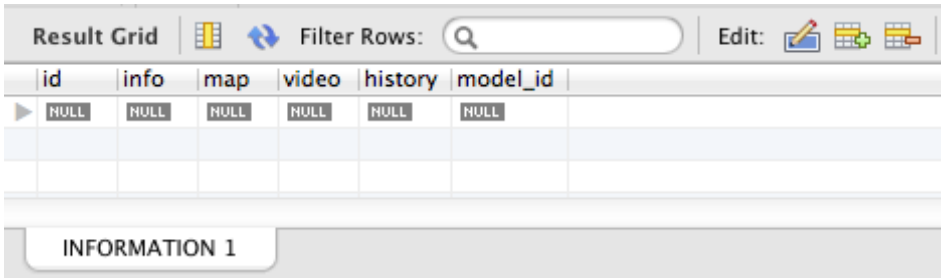
Εικόνα 5.12 - Πίνακας Pinsτης βάσης δεδομένων

Πίνακας Information:

```
-- -----------------------------------------------------
-- Table `crete3Ddb`.`Information`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `crete3Ddb`.`INFORMATION` (
  `id` INT(11) NOT NULL AUTO_INCREMENT,
  `info` TEXT ,
  `map`  VARCHAR(200) ,
  `video`  VARCHAR(200) ,
  `history` TEXT ,
  `model_id` INT(11),
  PRIMARY KEY (`id`),
  CONSTRAINT `INFO_BELONGS_TO_NODEL_ID` FOREIGN KEY (`model_id`)
  REFERENCES MODEL(id) ON DELETE NO ACTION ON UPDATE CASCADE
  );
```

Εικόνα 5.13 - MySQLδημιουργία πίνακα Information

Αποτέλεσμα στο γραφικό εργαλείο σχεδιασμού MySQLWorkbench:



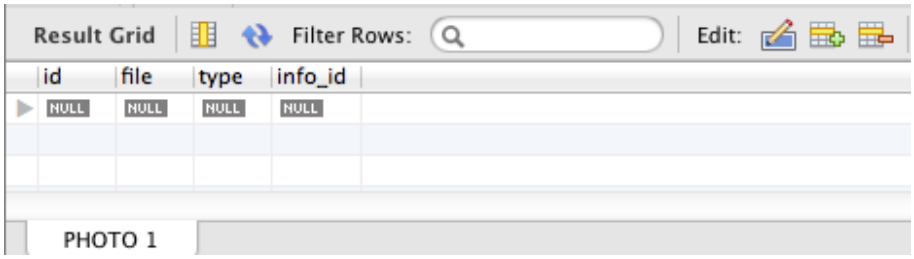
Εικόνα 5.14 - Πίνακας Informationτης βάσης δεδομένων

Πίνακας Photo:

```
-- -----------------------------------------------------
-- Table `crete3Ddb`.`Photo`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `crete3Ddb`.`PHOTO` (
  `id` INT(11) NOT NULL AUTO_INCREMENT,
  `file` BLOB ,
  `type`  VARCHAR(100) ,
  `info_id` INT(11),
  PRIMARY KEY (`id`),
  CONSTRAINT `INFO_BELONGS_TO_NODEL_ID` FOREIGN KEY (`info_id`)
  REFERENCES INFORMATION(id) ON DELETE NO ACTION ON UPDATE CASCADE
  );
```

Εικόνα 5.15 - MySQLδημιουργία πίνακα Photo

Αποτέλεσμα στο γραφικό εργαλείο σχεδιασμού MySQLWorkbench:



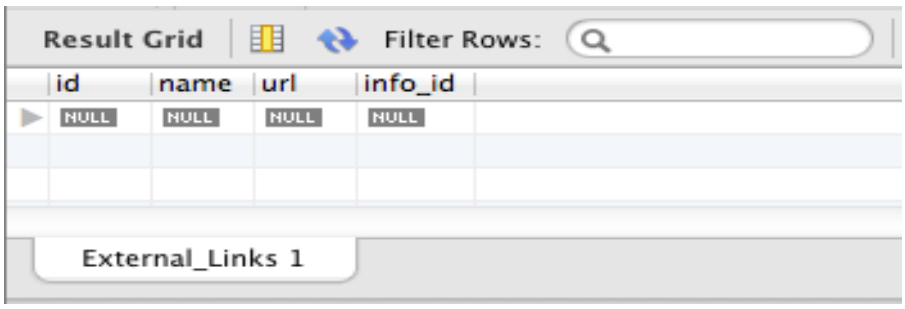
Εικόνα 5.16- Πίνακας Photoτης βάσης δεδομένων

Πίνακας External Links:

```
-- -----------------------------------------------------
-- Table `crete3Ddb`.`External_links`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `crete3Ddb`.`External_Links` (
  `id` INT(11) NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(100) ,
  `url`  VARCHAR(300) ,
  `info_id` INT(11),
  PRIMARY KEY (`id`),
  CONSTRAINT `INFO_BELONGS_TO_NODEL_ID` FOREIGN KEY (`info_id`)
  REFERENCES INFORMATION(id) ON DELETE NO ACTION ON UPDATE CASCADE
  );
```

Εικόνα 5.17 – MySQLδημιουργία πίνακα ExternalLinks

Αποτέλεσμα στο γραφικό εργαλείο σχεδιασμού MySQLWorkbench:



Εικόνα 5.18- Πίνακας ExternalLinksτης βάσης δεδομένων

Δημιουργήσαμε λοιπόν την βάση δεδομένων που χρειαζόμαστε με τον κώδικα που δείξαμε παραπάνω, και είναι έτοιμη να δεχθεί δεδομένα για αποθήκευση. Τα δεδομένα αυτά τα εισαγάγαμε μέσω του εργαλείου διαχείρισης της εφαρμογής που έχουμε αναφέρει σε προηγούμενα κεφάλαια.

5.3 Μεριά διακομιστή (Server Side)

Έχοντας σχεδιάσει και δημιουργήσει την βάση δεδομένων για την εφαρμογή μας, γράψαμε τον κώδικα ο οποίος αναπαριστά το businessLogicκαι tocontroller-servicesεπίπεδο του server-sidetης αρχιτεκτονικής που επεξηγήθηκε στην ενότητα 3.4. Χρησιμοποιήσαμε το ολοκληρωμένο περιβάλλον NetBeansIDEγια την υλοποίηση αυτού του μέρους. Δημιουργήσαμε λοιπόν στο περιβάλλον αυτό ένα WebApplicationProject, και εν συνεχεία στο πακέτο sourcesδημιουργήσαμε τους φακέλους που ήταν απαραίτητοι για την σωστή οργάνωση της εφαρμογής.Δημιουργήσαμε λοιπόν ένα φάκελο crete3d.db.daoστον οποίο δημιουργήσαμε την abstract κλάση DAOFactory (εικόνα 5.21)και τις αντίστοιχες abstractclassesγια τα απαραίτητα DAOobjectsέτσι ώστε να υπάρχει η δυνατότητα να μπορεί εύκολα να υλοποιηθεί το αντίστοιχο μοντέλο για διαφορετικές βάσεις

δεδομένων καθώς και για διαφορετικού τύπου βάσεις δεδομένων και όχι μόνο για MySQLπου έχουμε υλοποιήσει εμείς.Οι abstractκλάσεις ακολουθούν την ονοματολογία NameofClassDAOκαι οι υλοποιήσεις τους για MySQLτην ονοματολογία MySQLNameOfClassDAO. Για την διαχείριση των αντικειμένων που προκύπτουν από τις οντότητες της βάσης δεδομένων, δημιουργήσαμε τον φάκελο db.model. Σε αυτόν υπάρχουν τα αντικείμενα-κλάσεις που αντιστοιχούν στις οντότητες της βάσης δεδομένων. Τέλος δημιουργήσαμε τον φάκελο crete3d.server.model, στον οποίο έχουμε και πάλι κλάσεις- αντικείμενα, με την ονοματολογία MyClassNameBO, αντίστοιχα της βάσηςδεδομένων, αλλά με λιγότερα στοιχεία από τα db.model.

5.3.1 DataBase Model

Όπως αναφέραμε τα db.modelαφορούν τα αντικείμενα που δημιουργήσαμε τα οποία έχουν ως σκοπό τη δημιουργία στιγμιότυπων για επικοινωνία με την βάση δεδομένων.

```
1  public class Model {
2
3      private int modelId;
4      private String modelName;
5      private String objUrl;
6      private String mtlUrl;
7      private String clickable;
8      private String hover;
9      private String modelType;
10     private int groundId;
11     private int levelId;
12     private int periodId;
13     private float scale;
14
15     public Model() {
16
17         modelId = 0;
18         modelName = null;
19         modelType = null;
20         objUrl = null;
21         mtlUrl = null;
22         groundId = 0;
23         periodId = 0;
24         levelId = 0;
25         clickable = null;
26         hover = null;
27         scale = 1;
28
29
30     }
31     //there are getter and settors methods too
32 }
```

Εικόνα 5.19 – κλάση model του πακέτου db.model

Όπως φαίνεται και στον παραπάνω κώδικα (εικόνα 5.19) το αντικείμενο model, το οποίο αντιστοιχεί στην οντότητα modelτης βάσης δεδομένων, περιλαμβάνει όλα τα στοιχεία που έχει και το αντίστοιχο στην βάση. Αντίστοιχα όλα τα στοιχεία της οντότητας Pinstης βάσης δεδομένων περιέχει και το db.modelPins (εικόνα 5.20)

καθώς και οι υπόλοιπες κλάσεις που μοντελοποιούν την βάση δεδομένων Period, Level, Information, Photo, ExternalLinks.

```
1  public class Pins {
2
3      private int id;
4      private String name;
5      private String objUrl;
6      private String mtlUrl;
7      private String clickable;
8      private int ground_id;
9      private int points_id;
10     private int period_id;
11
12     public Pins() {
13         id = 0;
14         name = null;
15         objUrl = null;
16         mtlUrl = null;
17         clickable = null;
18         ground_id = 0;
19         points_id = 0;
20         period_id = 0;
21     }
22 }
```

Εικόνα 5.20 - κλάση Pins του πακέτου db.model

5.3.2 DAO (data access objects)

Έχοντας υλοποίηση στην παραπάνω ενότητα τα αντικείμενα-κλάσεις για όλες τις οντότητες της βάσης δεδομένων, επόμενο βήμα είναι η επικοινωνία τους με την βάση δεδομένων.

```
1  public abstract class DAOFactory {
2       public static final int MYSQL = 1;
3
4      public abstract ModelDAO getModelDAO();
5      public abstract PinsDAO getPinsDAO();
6      public abstract PeriodDAO getPeriodDAO();
7      public abstract LevelDAO getLevelDAO();
8      public abstract InformationDAO getInformationDAO();
9      public abstract PhotoDAO getPhotoDAO();
10     public abstract ExternalLinksDAO getExternalDAO();
11
12     public static DAOFactory getDAOFactory(int whichFactory) {
13         switch (whichFactory) {
14             case MYSQL:
15                 return new MySqlDAOFactory();
16             default:
17                 return null;
18         }
19     }
20 }
```

Εικόνα 5.21 – DAOFactory

Για την υλοποίηση της abstractκλάσης DAOFactory (εικόνα 5.21), δημιουργήσαμε την MySqlFactoryDAO (εικόνα 5.22), η οποία περιέχει τα απαραίτητα στοιχεία για τη σύνδεση με την MySQLβάση δεδομένων που υλοποιήσαμε, και την αντιστοίχηση

των abstract DAO objects με τα MySqlObjectDAO. Τα απαραίτητα στοιχεία είναι το urlτης σύνδεσης που δημιουργήσαμε στην προηγούμενη ενότητα και το όνομα χρήστη και τον κωδικό του για να επιτευχθεί η επικοινωνία με την βάση δεδομένων.

```
1  public class MySqlFactoryDAO extends DAOFactory{
2        public static final String DRIVER = "com.mysql.jdbc.Driver";
3
4      public static final String DBURL = "jdbc:mysql://localhost:3306/crete3Ddb";
5
6      public static Connection createConnection() {
7          Connection conn = null;
8
9          try {
10             Class.forName(DRIVER);
11             conn = DriverManager.getConnection(DBURL, "root", "root");
12         } catch (Exception e) {
13             e.printStackTrace();
14         }
15
16         return conn;
17     }
18
19     @Override
20     public ModelDAO getModelDAO() {
21         return new MySqlModelDAO();
22     }
23
24     @Override
25     public PinsDAO getPinsDAO() {
26         return new MySqlPinsDAO();
27     }
28 }
```

Εικόνα 5.22 – MySqlFactoryDAO

Για να γίνει πιο κατανοητή η έννοια των DAO, αν είχαμε για παράδειγμα δεδομένα και σε μία άλλου τύπου βάση δεδομένων όπως η MongoDB, θα είχαμε το αντίστοιχο MongoFactoryDAOκαθώς και τα αντίστοιχα MongoObjectDAO. Επομένως με αυτό τον τρόπο δεν δεσμεύεται η εφαρμογή μας να λειτουργεί μόνο με MySQLβάση δεδομένων, αλλά γράφοντας μόνο τα κομμάτια αυτά, των DAOObjects , μπορεί να λειτουργεί πάνω σε οποιαδήποτε τύπου βάση δεδομένων.

5.3.2.1 ModelDAO

```
1  public interface ModelDAO {
2
3      public int insertModel(Model model);
4
5      public boolean deleteModel(int modelId);
6
7      public Model getModel(int modelId);
8
9      public boolean updateModel(Model model);
10
11     public Model getModelbyName(String model_name);
12
13     public Model getGround(int groundId);
14
15     public List<Model> getGroundList();
16
17     public List<Model> getModelListByGround(int groundId);
18
19 }
```

Εικόνα 5.23 – ModelDAO

Το ModelDAOόπως βλέπουμε και στον κώδικα παραπάνω (εικόνα 5.23) ουσιαστικά περιλαμβάνει τις συναρτήσεις οι οποίες έχουν υλοποιηθεί από το MySQLModelDAO. Αναλυτική περιγραφή των συναρτήσεων:

1. PublicintinsertModel(Modelmodel) (εικόνα 5.24): Η συνάρτηση αυτή λαμβάνει σαν όρισμα ένα αντικείμενο τύπου Model(db.model) και καλείται για την εισαγωγή του ορίσματος αυτού στην βάση δεδομένων. Βλέπουμε και στην εικόνα ότι γίνεται η σύνδεση με την βάση δεδομένων και στην συνέχεια εκτελείται το sqlqueryπου θέλουμε να εκτελεστεί στην βάση δεδομένων.Η συνάρτηση αυτή επιστρέφει έναvint (δηλαδή ακέραιο αριθμό) ο οποίος ανάλογα με την τιμή του, μας δείχνει αν όλα πήγαν σωστά.

```java
@Override
    public int insertModel(Model model) {
        int result = 0;
        PreparedStatement pst = null;
        ResultSet generatedKeys = null;

        int affected = 0;
        Connection conn = MySqlDAOFactory.createConnection();

        try {
            pst = conn.prepareStatement("INSERT INTO MODEL( name, obj, mtl, clickable,"
                    + "hover, type, ground_id, level_id, period_id,scale) "
                    + "VALUES( ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)", Statement.RETURN_GENERATED_KEYS);

            pst.setString(1, model.getModelName());
            pst.setString(2, model.getObjUrl());
            pst.setString(3, model.getMtlUrl());
            pst.setString(4, model.getClickable());
            pst.setString(5, model.getHover());
            pst.setString(6, model.getModelType());

            if (model.getGroundId() == 0) {
                pst.setString(7, null);
            } else {
                pst.setInt(7, model.getGroundId());
            }

            if (model.getLevelId() == 0) {
                pst.setString(8, null);
            } else {
                pst.setInt(8, model.getLevelId());
            }

            if (model.getPeriodId() == 0) {
                pst.setString(9, null);
            } else {
                pst.setInt(9, model.getPeriodId());
            }

            pst.setFloat(10, model.getScale());

            pst.executeUpdate();

            generatedKeys = pst.getGeneratedKeys();
            if (generatedKeys.next()) {
                result = generatedKeys.getInt(1);
            } else {
                throw new DBOperationException("Error in insert Post");
            }

        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            try {
                conn.close();
            } catch (SQLException ex) {
                ex.printStackTrace();
            }
        }
        return result;
```

Εικόνα 5.24- Υλοποίηση της συνάρτησης InsertModel

2. Public boolean deleteModel(int modelId): Η συνάρτηση αυτή λαμβάνει σαν όρισμα το modelIdκαι επιχειρεί να διαγράψει από την βάση δεδομένων το στιγμιότυπο το οποίο έχει σαν πρωτεύον κλειδί αυτό το όρισμα. Αν όλα πάνε καλά τότε επιστρέφει true, ειδάλλως επιστρέφει false. Για αυτό και είναι τύπου Boolean.

```java
@Override
public boolean deleteModel(int modelId) {

    PreparedStatement pst = null;
    int affected = 0;

    Connection conn = MySqlDAOFactory.createConnection();

    try {
        conn.setAutoCommit(false);

        pst = conn.prepareStatement("DELETE FROM MODEL WHERE id = ?");
        pst.setInt(1, modelId);

        affected = pst.executeUpdate();

        if (affected == 1) {
            return true;
        } else {
            return false;
        }

    } catch (Exception e) {
        e.printStackTrace();
        return false;
    } finally {
        try {
            conn.close();
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
    }

}
```

Εικόνα 5.25 – Υλοποίηση της συνάρτησης deleteModel

3. publicModelgetModel (intmodelId): Η συνάρτηση αυτή καλείται για την ανάκτηση ενός στιγμιότυπου της οντότητας Modelαπό την βάση δεδομένων. Έχει σαν όρισμα της το modelIDτο οποίο είναι και πρωτεύον κλειδί της οντότητας αυτής. Αν βρεθεί το στιγμιότυπο με το συγκεκριμένο modelIdτότε επιστρέφει το στιγμιότυπο τύπου Model.(εικόνα 5.26)

```java
@Override
public Model getModel(int modelId) {

    PreparedStatement pst = null;
    Connection conn = MySqlDAOFactory.createConnection();

    Model model = new Model();

    try {
        pst = conn.prepareStatement("SELECT * FROM MODEL WHERE id = ?");
        pst.setInt(1, modelId);

        ResultSet rs = pst.executeQuery();

        while (rs.next()) {
            model.setModelId(rs.getInt("id"));
            model.setModelName(rs.getString("name"));
            model.setObjUrl(rs.getString("obj"));
            model.setMtlUrl(rs.getString("mtl"));
            model.setClickable(rs.getString("clickable"));
            model.setHover(rs.getString("hover"));
            model.setModelType(rs.getString("type"));
            model.setGroundId(rs.getInt("ground_id"));
            model.setLevelId(rs.getInt("level_id"));
            model.setPeriodId(rs.getInt("period_id"));
            model.setScale(rs.getFloat("scale"));

        }

    } catch (SQLException ex) {
        try {
            throw new DBOperationException(ex);
        } catch (DBOperationException ex1) {
            Logger.getLogger(MySqlModelDAO.class.getName()).log(Level.SEVERE, null, ex1);
        }
    } finally {
        try {
            conn.close();
        } catch (SQLException ex) {
            try {
                throw new DBOperationException(ex);
            } catch (DBOperationException ex1) {
                Logger.getLogger(MySqlModelDAO.class.getName()).log(Level.SEVERE, null, ex1);
            }
        }
    }

    return model;

}
```

Εικόνα 5.26 – Υλοποίηση της συνάρτησης getModel.

4. publicbooleanupdateModel(ModelmyModel): Ησυνάρτηση αυτή καλείται για την ανανέωση των δεδομένων ενός στιγμιότυπου της οντότητας Model της βάσης δεδομένων. Σαν όρισμα δέχεται το μοντέλο που έχει υποστεί τις αλλαγές., το οποίο όμως δεν γίνεται να αλλάξει το modelId, επομένως μέσω αυτού γίνεται η αναζήτηση του στην βάση δεδομένων. Επιστρέφει trueαν ολοκληρωθεί η ανανέωση, και falseαν όχι. (εικόνα 5.27)
5. publicModelgetModelByName(stringmodel_name): Ησυνάρτηση αυτή καλείται για την ανάκτηση ενός στιγμιότυπουModel της βάσης δεδομένων, με βάση το όνομα του. Παίρνει λοιπόν σαν όρισμα το όνομα του μοντέλου που αναζητείται στην βάση για ανάκτηση και αν βρεθεί το επιστρέφει σαν τύπου Model, ειδάλλως επιστρέφει null. (εικόνα 5.28)

```java
@Override
 public boolean updateModel(Model model) {

     PreparedStatement pst = null;
     int affected = 0;
     Connection conn = MySqlDAOFactory.createConnection();

     try {

         pst = conn.prepareStatement("UPDATE MODEL "
           |       + "SET id = ?, name = ?,obj = ?, mtl = ?, clickable = ?,"
                   +" hover = ?, type = ?, ground_id = ?, level_id = ?,"
                   +"period_id = ?, scale = ? "
                   + "WHERE id = ?");

         pst.setInt(1, model.getModelId());
         pst.setString(2, model.getModelName());
         pst.setString(3, model.getObjUrl());
         pst.setString(4, model.getMtlUrl());
         pst.setString(5, model.getClickable());
         pst.setString(6, model.getHover());
         pst.setString(7, model.getModelType());

         if (model.getGroundId() == 0) {
             pst.setString(8, null);
         } else {
             pst.setInt(8, model.getGroundId());
         }

         if (model.getLevelId() == 0) {
             pst.setString(9, null);
         } else {
             pst.setInt(9, model.getLevelId());
         }

         if (model.getPeriodId() == 0) {
             pst.setString(10, null);
         } else {
             pst.setInt(10, model.getPeriodId());
         }

         pst.setFloat(11, model.getScale());

         pst.setInt(12, model.getModelId());

         affected = pst.executeUpdate();

         if (affected == 1) {
             return true;
         } else {
             return false;
         }

     } catch (Exception e) {
         e.printStackTrace();
         return false;
     } finally {
         try {
             conn.close();
         } catch (SQLException ex) {
             ex.printStackTrace();
         }
     }
 }
```

Εικόνα 5.27 – Υλοποίηση της συνάρτησης UpdateModel

```java
@Override
public Model getModelbyName(String model_name) {

    PreparedStatement pst = null;
    Connection conn = MySqlDAOFactory.createConnection();

    Model model = new Model();

    try {
        pst = conn.prepareStatement("SELECT * FROM MODEL WHERE name = ?");
        pst.setString(1, model_name);

        ResultSet rs = pst.executeQuery();

        while (rs.next()) {
            model.setModelId(rs.getInt("id"));
            model.setModelName(rs.getString("name"));
            model.setObjUrl(rs.getString("obj"));
            model.setMtlUrl(rs.getString("mtl"));
            model.setClickable(rs.getString("clickable"));
            model.setHover(rs.getString("hover"));
            model.setModelType(rs.getString("type"));
            model.setGroundId(rs.getInt("ground_id"));
            model.setLevelId(rs.getInt("level_id"));
            model.setPeriodId(rs.getInt("period_id"));

        }

    } catch (SQLException ex) {
        try {
            throw new DBOperationException(ex);
        } catch (DBOperationException ex1) {
            Logger.getLogger(MySqlModelDAO.class.getName()).log(Level.SEVERE, null, ex1);
        }
    } finally {
        try {
            conn.close();
        } catch (SQLException ex) {
            try {
                throw new DBOperationException(ex);
            } catch (DBOperationException ex1) {
                Logger.getLogger(MySqlModelDAO.class.getName()).log(Level.SEVERE, null, ex1);
            }
        }
    }

    return model;

}
```

Εικόνα 5.28 – Υλοποίηση της συνάρτησης getModelByName

6. publicModelgetGround(intgroundId): Η συνάρτηση αυτή καλείται για την ανάκτηση ενός συγκεκριμένου ground. Ουσιαστικά είναι η ίδια με την getModel, αλλά εμπεριέχει και τον έλεγχο ότι το typetou στιγμιότυπου του Modelτης βάσης δεδομένων είναι «Ground». Επιστρέφει το μοντέλο που ανακτά, ειδάλλως null. (εικόνα 5.29)

```java
@Override
public Model getGround(int groundId) {

    PreparedStatement pst = null;
    Connection conn = MySqlDAOFactory.createConnection();

    Model model = new Model();

    try {
        pst = conn.prepareStatement("SELECT * FROM MODEL WHERE id = ? and Type= 'GROUND'");
        pst.setInt(1, groundId);

        ResultSet rs = pst.executeQuery();

        while (rs.next()) {
            model.setModelId(rs.getInt("id"));
            model.setModelName(rs.getString("name"));
            model.setObjUrl(rs.getString("obj"));
            model.setMtlUrl(rs.getString("mtl"));
            model.setClickable(rs.getString("clickable"));
            model.setHover(rs.getString("hover"));
            model.setModelType(rs.getString("type"));
            model.setGroundId(rs.getInt("ground_id"));
            model.setLevelId(rs.getInt("level_id"));
            model.setPeriodId(rs.getInt("period_id"));

        }

    } catch (SQLException ex) {
        try {
            throw new DBOperationException(ex);
        } catch (DBOperationException ex1) {
            Logger.getLogger(MySqlModelDAO.class.getName()).log(Level.SEVERE, null, ex1);
        }
    } finally {
        try {
            conn.close();
        } catch (SQLException ex) {
            try {
                throw new DBOperationException(ex);
            } catch (DBOperationException ex1) {
                Logger.getLogger(MySqlModelDAO.class.getName()).log(Level.SEVERE, null, ex1);
            }
        }
    }

    return model;

}
```

Εικόνα 5.29– Υλοποίηση της συνάρτησης getGround

7. publicList<Model>getModelListByGround(intgroundId): Η συνάρτηση αυτή καλείται για την ανάκτηση όλων στιγμιότυπωνModel της βάσης δεδομένων, τα οποία έχουν σαν foreignkeyτο groundIdτο οποίο έχει δοθεί σαν όρισμα στην συνάρτηση. Με τον τρόπο αυτό συλλέγουμε όλα εκείνα τα μοντέλα τα οποία βρίσκονται πάνω σε κάποιο συγκεκριμένο μοντέλο-έδαφος. Τα στιγμιότυπα επιστρέφονται σε λίστα από Model. (εικόνα 5.30)

```java
@Override
public List<Model> getModelListByGround(int groundId) {

    PreparedStatement pst = null;
    Connection conn = MySqlDAOFactory.createConnection();
    List<Model> model_list = new ArrayList<Model>();

    try {
        pst = conn.prepareStatement("SELECT * FROM MODEL WHERE ground_id = ?");
        pst.setInt(1, groundId);

        ResultSet rs = pst.executeQuery();

        while (rs.next()) {

            Model model = new Model();

            model.setModelId(rs.getInt("id"));
            model.setModelName(rs.getString("name"));
            model.setObjUrl(rs.getString("obj"));
            model.setMtlUrl(rs.getString("mtl"));
            model.setClickable(rs.getString("clickable"));
            model.setHover(rs.getString("hover"));
            model.setModelType(rs.getString("type"));
            model.setGroundId(rs.getInt("ground_id"));
            model.setLevelId(rs.getInt("level_id"));
            model.setPeriodId(rs.getInt("period_id"));
            model.setScale(rs.getFloat("scale"));

            model_list.add(model);

        }

    } catch (SQLException ex) {
        try {
            throw new DBOperationException(ex);
        } catch (DBOperationException ex1) {
            Logger.getLogger(MySqlModelDAO.class.getName()).log(Level.SEVERE, null, ex1);
        }
    } finally {
        try {
            conn.close();
        } catch (SQLException ex) {
            try {
                throw new DBOperationException(ex);
            } catch (DBOperationException ex1) {
                Logger.getLogger(MySqlModelDAO.class.getName()).log(Level.SEVERE, null, ex1);
            }
        }
    }

    return model_list;
}
```

Εικόνα 5.30 – Υλοποίηση της συνάρτησης getModelListByGround

5.3.2.2 PinsDAO

```
1  public interface PinsDAO {
2
3      public int insertPin(Pins pin);
4
5      public Pins getPinByName(String name);
6
7      public List<Pins> getModelPins(int modelId);
8
9      public Pins getPinById(int id);
10
11     public List<Pins> getPeriodPins(int periodId);
12
13     public List<Pins> getNonPointingPeriodPins(int periodId);
14
15     public boolean updatePinsPointsTo(String pinId, int pointsId);
16
17     public List<Pins> getPinsByGround(int groundId);
18
19 }
```

Εικόνα 5.31- PinsDAO

Στο PinsDAO όπως και στο ModelDAO που εξετάσαμε προηγουμένως, δηλώνονται οι συναρτήσεις οι οποίες βλέπουμε να υλοποιούνται για την περίπτωση μας στο MySQLPinsDAO. Οκώδικας είναι αντίστοιχος με των ModelDAO, οπότε θα περιγράψουμε την λειτουργία της κάθε συνάρτησης:

1. public int insertPin(Pins pin): Ησυνάρτηση αυτή καλείται για την εισαγωγή στη βάση δεδομένων ενός στιγμιότυπου αντικειμένου Pins(του φακέλου db.model). Παίρνει σαν όρισμα λοιπόν το στιγμιότυπο αυτό, και το εισάγει στην βάση δεδομένων στην οντότητα Pins. Αν η εισαγωγή γίνει επιτυχώς, τότε επιστρέφεται το pinId το οποίο παράχθηκε από την βάση δεδομένων.
2. public Pins getPinByName(String name): Η συνάρτηση αυτή καλείται για την ανάκτηση ενόςστιγμιότυπουPinsαπό την βάση δεδομένων, το οποίο έχει σαν όνομα «name». Το όνομα περνά στην συνάρτηση σαν όρισμα, και αν βρεθεί τέτοιο στιγμιότυπο στην βάση δεδομένων, τότε ανακτάτε και επιστρέφεται ως στιγμιότυπο αντικειμένου Pins.
3. publicList<Pins>getModelPins(intmodelId): Ησυνάρτηση αυτή καλείται για την ανάκτηση όλων εκείνων των pinsτα οποία ανήκουν σε κάποιο συγκεκριμένο μοντέλο. Δηλαδή ανακτά τα στιγμιότυπα Pinsτης βάσης δεδομένων, τα οποία έχουν σαν foreignkey (belongs_id) το όρισμα της συνάρτησης modelId. Αν βρεθούν αποτελέσματα, τότε τα στιγμιότυπα των Pinsεπιστρέφονται στην μορφή λίστας τύπου Pins.
4. publicPinsgetPinById(intid): Η συνάρτηση αυτή καλείται για την ανάκτηση ενός στιγμιότυπουPins της βάσης δεδομένων, το οποίο έχει σαν πρωτεύον κλειδί το idτο οποίο περνά στην συνάρτηση σαν όρισμα. Αν βρεθεί το στιγμιότυπο τότε επιστρέφεται σαν στιγμιότυπο αντικειμένου Pins.
5. publicList<Pins>getPeriodPins(intperiodId): Η συνάρτηση αυτή καλείται για την ανάκτηση όλων εκείνων των στιγμιότυπωνPinsτης βάσης δεδομένων τα οποία έχουν σαν foreignkeyτο όρισμα της συνάρτησης periodId. Επιστρέφονται με μορφή λίστας από αντικείμενα Pins.
6. publicList<Pins>getNonPointingPeriodPins(intperiodId): Η συνάρτηση αυτή καλείται για την ανάκτηση όλων εκείνων των στιγμιότυπωνPinsτης βάσης

δεδομένων, τα οποία δεν έχουν ορισμένο foreignkey “points_id”, δηλαδή δεν δείχνουν πουθενά. Επιστρέφονται και αυτά σε λίστα αντικειμένων Pins.

7. publicbooleanupdatePinsPointsTo(StringpinId, intpointsId): Ησυνάρτηση αυτή καλείται για την ανανέωση ενός συγκεκριμένου στοιχείου του στιγμιότυπουPinsτης βάσης δεδομένων. Ουσιαστικά καλείται για να θέσει κάποιο pin που έχει ήδη δημιουργηθεί να δείχνει σε κάποιο μοντέλο. Δέχεται σαν όρισμα το pinIdτου στιγμιότυπου που θέλει να αλλάξει, και το pointsId το οποίο είναι το στοιχείο το οποίο θέλει να αλλάξει στο στιγμιότυπο.
8. publicList<Pins>getPinsByGround(intgroundId): Η συνάρτηση αυτή καλείται για την ανάκτηση όλων εκείνων των στιγμιότυπωνPinsτης βάσης δεδομένων τα οποία ανήκουν σε συγκεκριμένο στιγμιότυπο Model που είναι τύπου «Ground». Δέχεται σαν όρισμα το groundIdκαι τα στιγμιότυπα τύπου Pinsτα οποία έχουν foreignkeyτο όρισμα αυτό, επιστρέφονται σε λίστα αντικειμένων Pins.

5.3.2.3 PeriodDAO

```
1  public interface PeriodDAO {
2
3      public Period getPeriod(int periodId);
4
5      public List<Period> getAllPeriod();
6
7  }
```

Εικόνα 5.32- PeriodDAO

Οι συναρτήσεις για το periodDAO:

1. public Period getPeriod(int periodId): Η συνάρτηση αυτή καλείται για να μας επιστρέψει το στιγμιότυπο Periodτης βάσης δεδομένων το οποίο έχει πρωτεύον κλειδί το periodId που έχει σαν όρισμα η συνάρτηση.
2. public List<Period> getAllPeriod(): Η συνάρτηση αυτή καλείται για την ανάκτηση όλων των στιγμιότυπων Period οι οποίες υπάρχουν στη βάση δεδομένων μας και επιστρέφονται σε λίστα αντικειμένου Period.

5.3.2.4 LevelDAO

```
1  public interface LevelDAO {
2
3      public Level getLevel(int levelId);
4
5      public List<Level> getAllLevel();
6
7  }
```

Εικόνα 5.33 –LevelDAO

Οι συναρτήσεις για το LevelDAO:

1. public Level getLevel(int levelId): Η συνάρτηση αυτή καλείται για να μας επιστρέψει το στιγμιότυπο Levelτης βάσης δεδομένων το οποίο έχει πρωτεύον κλειδί το levelId που έχει σαν όρισμα η συνάρτηση. Επιστρέφει ένα αντικείμενο Level (του πακέτου db.model).

2. public List<Level> getAllLevel():Η συνάρτηση αυτή καλείται για την ανάκτηση όλων των στιγμιότυπων Level οι οποίες υπάρχουν στη βάση δεδομένων μας και επιστρέφονται σε λίστα αντικειμένου Level.

5.3.2.5 InformationDAO

```
1  public interface InformationDAO {
2
3     public Information getInformation(int modelId);
4
5   }
```

Εικόνα 5.34 – InformationDAO

Η συνάρτηση που υλοποιήσαμε για το LevelDAO:

public Information getInformation(int modelId): Η συνάρτηση αυτή καλείται για να μας επιστρέψει το στιγμιότυπο Informationτης βάσης δεδομένων το οποίο έχει ως foreignkeyτο modelId που έχει σαν όρισμα η συνάρτηση. Επιστρέφει ένα αντικείμενο Information (του πακέτου db.model).

5.3.3 Business Objects

Στις προηγούμενες ενότητες αναφερθήκαμε στην υλοποίηση των DAOκαι των db.modelτα οποία είναι απαραίτητα για την επικοινωνία με την βάση δεδομένων. Συνδέουμε λοιπόν με αυτόν τον τρόπο την βάση δεδομένων και το server-sideτης εφαρμογής. Χρειαζόμαστε όμως και μία σύνδεση με τον clientκαι πιο συγκεκριμένα «αντικείμενα» επικοινωνίας του serverμε τον clientκαι αντίστροφα. Για τον λόγο αυτό για τα αντικείμενα που δημιουργήσαμε στο πακέτο db.model, δημιουργήσαμε τα αντίστοιχα αντικείμενα στο server.modelπακέτο. Αυτά τα αντικείμενα-κλάσεις ακολουθούν την εξής ονοματολογία: MyClassNameBO. Ως εκ τούτου, δημιουργήσαμε τις παρακάτω κλάσεις-αντικείμενα, οι οποίες θα αναλυθούν και περαιτέρω: ModelBO, PinsBO, PeriodBO, LevelBO, InformationBO.

Τα αντικείμενα αυτά, έχουν τις μεταβλητές οι οποίες θέλουμε να επιτρέπουμε να φτάνουν στον χρήστη, καθώς και τις μεθόδους-συναρτήσεις οι οποίες μας είναι απαραίτητες για το κάθε αντικείμενο ξεχωριστά. Να τονίσουμε σε αυτό το σημείο ότι για την επικοινωνία μεταξύ τωνDAOκαι των BO δημιουργήθηκαν οι απαραίτητες μέθοδοι η υλοποίηση των οποίων έγινε στις κλάσεις των DAO όπως για παράδειγμα η μέθοδος toModelBO() της οποίας το σώμα βρίσκεται στην κλάση ModelDAO. Όπως φαίνεται και στον κώδικα, γίνεται η αντιστοίχηση των μεταβλητών οι οποίες θέλουμε να περαστούν στα BOαντικείμενα. Υλοποιήθηκαν με τον ιδιο τρόπο λοιπόν οι μέθοδοι: toPinsBO, toPeriodBO, toLevelBO, toInformationBO.

```
public ModelBO toModelBO(){

        ModelBO modelBO = new ModelBO();

        modelBO.setId(modelId);
        modelBO.setName(modelName);
        modelBO.setType(modelType);
        modelBO.setObjUrl(objUrl);
        modelBO.setMtlUrl(mtlUrl);
        modelBO.setPeriodId(periodId);
        modelBO.setGroundId(groundId);
        modelBO.setLevelId(levelId);
        modelBO.setClickable(clickable);
        modelBO.setHover(hover);
        modelBO.setScale(scale);

        return modelBO;
    }
```

Εικόνα 5.35 – Υλοποίηση της μεθόδου toModelBO()

Όπως βλέπουμε στην εικόνα-5.35 υλοποιήθηκε η κλάσηModelBO η οποία έχει αντίστοιχα στοιχεία-μεταβλητές με το αντικείμενο ModelDAO. Επιπλέον υλοποιήθηκε η μέθοδος η οποία καλείται για την μετατροπή του αντικειμένου Javaσε JSONαντικείμενο (εικόνα 5.36) και μία μέθοδος η οποία από JSON αντικείμενο το μετατρέπει σε Javaαντικείμενο(εικόνα 5.37), οι οποίες χρησιμοποιήθηκαν για την αποστολή των δεδομένων μέσω της τεχνολογίας AJAX.

```
public class ModelBO {

    private int id;
    private String name;
    private String type;
    private String objUrl;
    private String mtlUrl;
    private int groundId;
    private int periodId;
    private int levelId;
    private String clickable;
    private String hover;
    private float scale;
    private int points_to;

    public ModelBO() {
        id = 0;
        name = null;
        type = null;
        objUrl = null;
        mtlUrl = null;
        groundId = 0;
        periodId = 0;
        levelId = 0;
        clickable = null;
        hover = null;
        scale =  1;
        points_to = 0;

    }
    // There are getters
    // and setters moethods too
}
```

Εικόνα 5.36 – Υλοποίηση της κλάσης ModelBO.

```
public String modelBoToJson() {
        String modelBoJson = null;

        modelBoJson = "{"
                + "\"id\"  : "
                + "\"" + this.getId() + "\"" + ","
                + "\"name\"  : "
                + "\"" + this.getName() + "\"" + ","
                + "\"type\"  : "
                + "\"" + this.getType() + "\"" + ","
                + "\"objUrl\"  : "
                + "\"" + this.getObjUrl() + "\"" + ","
                + "\"mtlUrl\"  : "
                + "\"" + this.getMtlUrl() + "\"" + ","
                + "\"groundId\"  : "
                + "\"" + this.getGroundId() + "\"" + ","
                + "\"periodId\"  : "
                + "\"" + this.getPeriodId() + "\"" + ","
                + "\"hover\"  : "
                + "\"" + this.getHover() + "\"" + ","
                + "\"clickable\"  : "
                + "\"" + this.getClickable() + "\"" + ","
                + "\"points_to\"  : "
                + "\"" + this.getPoints_to() + "\"" + ","
                + "\"scale\"  : "
                + "\"" + this.getScale() + "\"" + ","
                + " \"levelId\" : "
                + "\"" + this.getLevelId() + "\""
                + "}";

        return modelBoJson;
    }
```

Εικόνα 5.37- Υλοποίηση της μεθόδου modelBoToJson()

```
public ModelBO fromJson(String json){
       JSONObject json_obj = new JSONObject(json);

       String name = json_obj.getString("name");
       String type = json_obj.getString("type");
       String obj = json_obj.getString("obj");
       String mtl = json_obj.getString("mtl");
       String clickable = json_obj.getString("clickable");
       String hover = json_obj.getString("hover");
       int groundId = (int) json_obj.getInt("groundId");
       int leveLId = (int) json_obj.getInt("levelId");
       int periodId = (int) json_obj.getInt("periodId");
       float scale = Float.parseFloat(json_obj.getString("scale"));
       int comeFrom = (int) json_obj.getInt("comeFromPin");
       int id = (int)json_obj.getInt("id");

       ModelBO modelBO = new ModelBO();
       modelBO.setId(id);
       modelBO.setName(name);
       modelBO.setObjUrl(obj);
       modelBO.setMtlUrl(mtl);
       modelBO.setClickable(clickable);
       modelBO.setHover(hover);
       modelBO.setType(type);
       modelBO.setGroundId(groundId);
       modelBO.setLevelId(leveLId);
       modelBO.setPeriodId(periodId);
       modelBO.setScale(scale);

        return modelBO;
   }
```

Εικόνα 5.38 – Υλοποίηση της μεθόδου fromJson()

Αντίστοιχα με την υλοποίηση της κλάσης ModelBO, υλοποιήθηκαν και οι κλάσεις PinsBO, LevelBO, PeriodBO, InformationBOοι οποίες έχουν και τις αντίστοιχες μεθόδους μετατροπής τους σε JSONαντικείμενα και μετατροπής τους σε BusinessObjectαντικείμενα από JSON.

5.4 API

Έχοντας υλοποιήσει το μοντέλο της εφαρμογής μας στην πλευρά του διακομιστή αποφασίσαμε η επικοινωνία μεταξύ του διακομιστή και του χρήστη να πραγματοποιηθεί με την υλοποίηση ενός WEBAPI (ApplicationProgrammingInterface) το οποίο παρέχει υπηρεσίες RestfulServices. Τα δεδομένα που υποστηρίζει η εφαρμογή μας μέσω των υπηρεσιών που παρέχει είναι τύπου JSON, αλλά είναι παραμετροποιημένα με τέτοιον τρόπο ώστε εύκολα να υλοποιηθούν και για άλλου τύπου δεδομένα. Παρέχουμε CRUD RestfulServicesτα οποία παρέχουν υπηρεσίες για την δημιουργία, την ανάκτηση, την ανανέωση και την διαγραφή μοντέλων (Create, Retrieve, Update, Delete). Δημιουργήσαμε τρείς κατηγορίες των υπηρεσιών, οι οποίες είναι:

- “Model”: παρέχονται υπηρεσίες οι οποίες αφορούν το αντικείμενο Model
- “Pins”: Παρέχονται υπηρεσίες οι οποίες αφορούν την οντότητα Pins
- “Scene”: παρέχονται υπηρεσίες ανάκτησης των δεδομένων που φτάνουν στον χρήστη-επισκέπτη.

Οι υπηρεσίες που κατηγοριοποιήσαμε παραπάνω χρησιμοποιήθηκαν για την υλοποίηση της εφαρμογής διαχείρισης, αλλά και την κεντρική εφαρμογή του επισκέπτη. Πιο συγκεκριμένα, οι υπηρεσίες Model, Pinsχρησιμοποιήθηκαν για την παροχή υπηρεσιών εισαγωγής ανάκτησης και ανανέωσης των οντοτήτων αυτών από την σελίδα διαχείρισης. Ενώ η υπηρεσίες Sceneαφορούν την ανάκτηση όλων των πληροφοριών που είναι απαραίτητες να εμφανιστούν στον χρήστη ανάλογα με την διαδραστικότητα του.

5.4.1 Model

Στην κατηγορία Modelπαρέχουμε υπηρεσίες εισαγωγής (POST), ανάκτησης (GET), ανανέωσης (PUT) τρισδιάστατου μοντέλου (Model) για την εφαρμογή διαχείρισης. Δημιουργήσαμε λοιπόν μία κλάση ModelRestServiceκαι της δώσαμε το path “/model” και μέσα σε αυτήν υλοποιήσαμε τις μεθόδους get, post, put(εικόνα 5.39).

```java
@Path("/model")
public class ModelRESTService {

    @GET
    @Path("/get/{id}")
    @Produces({MediaType.APPLICATION_JSON})
    public String getModelJSON(@PathParam("id") String modelId) throws IOException {

        ModelBO result = null;

        try {
            result = getModel(Integer.parseInt(modelId));

        } catch (Exception ex) {
            ex.printStackTrace();
        }
        return result.modelBoToJson();
    }
}
```

Εικόνα 5.39 – Υλοποίηση ModelRestServiceκαι της μεθόδου-servicegetModelJson

Ημέθοδος getModelJson(@PathParam(“id”) StringmodelId) (εικόνα 5.39) λαμβάνει σαν όρισμα το modelIdτου τρισδιάστατου μοντέλου που θέλει ο χρήστης διαχειριστής να ανακτήσει από την βάση δεδομένων. Για την υλοποίηση αυτή καλείται η μέθοδος getModel() της οποίας το σώμα υλοποιήθηκε στην κλάση modelRestService (εικόνα 5.40) και καλώντας το απαραίτητο ModelDAOκαι την μέθοδο που έχουμε υλοποιήσει σε αυτό modelDAO.getModel() και τις απαραίτητες μεθόδους μετατροπής του σε modelBO.

```java
public ModelBO getModel(int modelId) {
        try {
            DAOFactory mySqlFactory = DAOFactory.getDAOFactory(DAOFactory.MYSQL);
            ModelDAO modelDAO = mySqlFactory.getModelDAO();

            ModelBO modelBO = new ModelBO();

            Model model = modelDAO.getModel(modelId);

            modelBO = model.toModelBO();

            return modelBO;
        } catch (Exception e) {
            return null;
        }
    }

```

Εικόνα 5.40 – Υλοποίηση της μεθόδου getModel

Η μέθοδος insertModelFromJson(Stringmymodel_json) λαμβάνει σαν όρισμα το αντικείμενο modelτύπου JSONπου έχει αποσταλεί από τον χρήστη-διαχειριστή, και αφού μετατραπεί σε Java αντικείμενο modelBO, καλείται η συνάρτηση insertModel() (εικόνα 5.41)η οποία δέχεται σαν όρισμα το modelBO, και αφού το μετατρέψει σε model του πακέτου model.db, καλεί την συνάρτηση modelDAO.insertModel() η οποία εισάγει το αντικείμενο στην βάση δεδομένων. Αν το στιγμιότυπο αυτό έχει τιμή στην μεταβλητή comeFrom, τότε καλείται η συνάρτηση η οποία είναι υπεύθυνη για την ανανέωση του αντικειμένου PINστην βάση δεδομένων, ώστε να ανανεωθεί η τιμή που έχει το foreignkeytου PINώστε να «δείχνει» στο μοντέλο που εισάγεται στην βάση μέσω της μεθόδου POST (εικόνα 5.42).

```
 1   public int InsertModel(ModelBO modelBO) throws DBOperationException {
 2         DAOFactory mySqlFactory = DAOFactory.getDAOFactory(DAOFactory.MYSQL);
 3         ModelDAO myModelDAO = mySqlFactory.getModelDAO();
 4
 5         Model mymodel = new Model();
 6         mymodel.setModelName(modelBO.getName());
 7         mymodel.setModelType(modelBO.getType());
 8         mymodel.setObjUrl(modelBO.getObjUrl());
 9         mymodel.setMtlUrl(modelBO.getMtlUrl());
10         mymodel.setGroundId(modelBO.getGroundId());
11         mymodel.setLevelId(modelBO.getLevelId());
12         mymodel.setPeriodId(modelBO.getPeriodId());
13         mymodel.setClickable(modelBO.getClickable());
14         mymodel.setHover(modelBO.getHover());
15         mymodel.setScale(modelBO.getScale());
16         int result = myModelDAO.insertModel(mymodel);
17
18         return result;
19     }
```

Εικόνα 5.41 – Υλοποίηση μεθόδου InsertModel(ModelBO modelBO)

```
 1  public boolean updatePointsToPin( String pinId, int pointsId ) {
 2         try {
 3             DAOFactory mySqlFactory = DAOFactory.getDAOFactory(DAOFactory.MYSQL);
 4             PinsDAO myPinsDAO = mySqlFactory.getPinsDAO();
 5
 6             boolean result = myPinsDAO.updatePinsPointsTo(pinId, pointsId);
 7
 8             return result;
 9         } catch (Exception e) {
10             return false;
11         }
12     }
```

Εικόνα 5.42 – Υλοποίηση μεθόδου updatePointsToPin()

```
@POST
    @Consumes({MediaType.APPLICATION_JSON})
    @Produces({MediaType.APPLICATION_JSON})
    public String insertModelfromJson(String mymodel_json1) {
        boolean result = false;
        ModelBO previewModel = null;

        JSONObject json_obj = new JSONObject(mymodel_json1);
        String name = json_obj.getString("name");
        String type = json_obj.getString("type");
        String obj = json_obj.getString("obj");
        String mtl = json_obj.getString("mtl");
        String clickable = json_obj.getString("clickable");
        String hover = json_obj.getString("hover");
        int groundId = (int) json_obj.getInt("groundId");
        int leveLId = (int) json_obj.getInt("levelId");
        int periodId = (int) json_obj.getInt("periodId");
        float scale = Float.parseFloat(json_obj.getString("scale"));
        String comeFrom =  json_obj.getString("comeFromPin");

        ModelBO modelBO = new ModelBO();
        modelBO.setName(name);
        modelBO.setObjUrl(obj);
        modelBO.setMtlUrl(mtl);
        modelBO.setClickable(clickable);
        modelBO.setHover(hover);
        modelBO.setType(type);
        modelBO.setGroundId(groundId);
        modelBO.setLevelId(leveLId);
        modelBO.setPeriodId(periodId);
        modelBO.setScale(scale);

        try {

            int generatedId = InsertModel=(modelBO);

            if(comeFrom != "" ){
               boolean pinUpdate = false;
               pinUpdate = updatePointsToPin( comeFrom, generatedId );

            }

            previewModel = getModel(generatedId);
            return previewModel.modelBoToJson();

             } catch (Exception ex) {
                 ex.printStackTrace();
             }

            return null;

    }
```

Εικόνα 5.43 – Υλοποίηση της μεθόδου-Service InsertModelFromJson()

Η μέθοδος updateModelFromJSON(stringjson_form) (εικόνα 5.44), λαμβάνει σαν όρισμα το αντικείμενο modelτύπου JSON από τον χρήστη διαχειριστή και αφού το μετατρέψει σε modelBO αντικείμενο, καλεί την συνάρτηση updateModel(ModelBOmodelBO) (εικόνα 5.45) η οποία μετατρέπει το modelBοσε Modelαντικείμενο το οποίο μέσω της μεθόδου modelDAO.updateModel(model) ανανεώνεται στην βάση δεδομένων.

```
@PUT
  @Consumes(MediaType.APPLICATION_JSON)
  @Produces(MediaType.APPLICATION_JSON)
  public String updateModelFROMJSON(String json_form) throws IOException {
      String result=null ;

      JSONObject json_obj = new JSONObject(json_form);
      String name = json_obj.getString("name");
      String type = json_obj.getString("type");
      String obj = json_obj.getString("obj");
      String mtl = json_obj.getString("mtl");
      String clickable = json_obj.getString("clickable");
      String hover = json_obj.getString("hover");
      int groundId = (int) json_obj.getInt("groundId");
      int leveLId = (int) json_obj.getInt("levelId");
      int periodId = (int) json_obj.getInt("periodId");
      float scale = Float.parseFloat(json_obj.getString("scale"));
      String comeFrom =  json_obj.getString("comeFromPin");
      int id= (int) json_obj.getInt("id");

      ModelBO modelBO = new ModelBO();
      modelBO.setName(name);
      modelBO.setObjUrl(obj);
      modelBO.setMtlUrl(mtl);
      modelBO.setClickable(clickable);
      modelBO.setHover(hover);
      modelBO.setType(type);
      modelBO.setGroundId(groundId);
      modelBO.setLevelId(leveLId);
      modelBO.setPeriodId(periodId);
      modelBO.setScale(scale);
      modelBO.setId(id);

      try {
         modelBO = updateModel(modelBO);
          result = modelBO.modelBoToJson();
           if(comeFrom != "" ){
             boolean pinUpdate = false;
             pinUpdate = updatePointsToPin( comeFrom, modelBO.getId() );

          }
      } catch (Exception ex) {
          ex.printStackTrace();
      }

      if (result != null) {
          return result;
      } else {
          return "{\n"
                  + "  \"success\" : " + result + ",\n"
                  + "  \"message\" : \"Updated User successfully\"\n"
                  + "}";
      }
  }
```

Εικόνα 5.44 – Υλοποίηση της μεθόδου-service updateModelFromJSON()

```
public ModelBO updateModel(ModelBO modelBO) {
        try {
            DAOFactory mySqlFactory = DAOFactory.getDAOFactory(DAOFactory.MYSQL);
            ModelDAO myModelDAO = mySqlFactory.getModelDAO();

            Model mymodel = new Model();
            mymodel.setModelId(modelBO.getId());
            mymodel.setModelName(modelBO.getName());
            mymodel.setModelType(modelBO.getType());
            mymodel.setObjUrl(modelBO.getObjUrl());
            mymodel.setMtlUrl(modelBO.getMtlUrl());
            mymodel.setGroundId(modelBO.getGroundId());
            mymodel.setLevelId(modelBO.getLevelId());
            mymodel.setPeriodId(modelBO.getPeriodId());
            mymodel.setClickable(modelBO.getClickable());
            mymodel.setHover(modelBO.getHover());
            mymodel.setScale(modelBO.getScale());
            boolean result = myModelDAO.updateModel(mymodel);

            modelBO = getModel(mymodel.getModelId());

            return modelBO;
        } catch (Exception e) {
            return null;
        }
    }
```

Εικόνα 5.45 – Υλοποίηση μεθόδου updateModel(ModelBOmyModelBO)

5.4.2 Pins

Στην κατηγορία Pins παρέχουμε υπηρεσίες για τον χρήστη-διαχειριστή οι οποίες αφορούν στην εισαγωγή Pinστην βάση δεδομένων, και την ανάκτησή τους από αυτή. Δημιουργήσαμε λοιπόν μία κλάση η οποία αφορά στο PinService που παρέχουμε η οποία λέγεται PinsRestService(εικόνα 5.46).

```java
@Path("/pins")
public class PinsRESTService {

    @POST
    @Consumes({MediaType.APPLICATION_JSON})
    @Produces({MediaType.APPLICATION_JSON})
    @Path("/insertPin")
    public String createPinFromJson(String json_pin) {
        int result = 0;
        ModelBO previewModel = null;
        PinsBO myPinBo = new PinsBO();
        myPinBo = myPinBo.fromJson(json_pin);

        try {
            result = insertPin(myPinBo);
            myPinBo=getPin(result);
        } catch (Exception ex) {
            ex.printStackTrace();
        }

        if (result == 0) {
            return "{\"model_id\":" + "\"" + result + "\","
                    + "\"model_name\":" + "\""  + "\"" + "}";
        } else {
            return myPinBo.pinBoToJson(myPinBo);
        }
    }
}
```

Εικόνα 5.46 – Υλοποίηση PinsRestServiceκαι της μεθόδου-service createPinFromJSON

Η μέθοδος createPinFromJSON(Stringjson_pin) (εικόνα 5.46) λαμβάνει σαν όρισμα από τον χρήστη διαχειριστή το αντικείμενο Pin τύπου JSONτο οποίο μετατρέπουμε σε αντικείμενο PinsBOκαι το εισάγουμε στην βάση δεδομένων μέσω της μεθόδου insertPin(myPinBO) (εικόνα 5.47)η οποία μετατρέπει το αντικείμενο pinBOσε Pinτο οποίο μέσω της μεθόδου pinDAO.insertPin(myPin) εισάγεται στην βάση δεδομένων της εφαρμογής.

```java
public int insertPin(PinsBO pinBO) throws DBOperationException {
       DAOFactory mySqlFactory = DAOFactory.getDAOFactory(DAOFactory.MYSQL);
       PinsDAO myPinDAO = mySqlFactory.getPinsDAO();

       Pins myPin = new Pins();
       myPin.setName(pinBO.getName());
       myPin.setObjUrl(pinBO.getObjUrl());
       myPin.setMtlUrl(pinBO.getMtlUrl());
       myPin.setClickable(pinBO.getClickable());
       myPin.setGround_id(pinBO.getGround_id());
       myPin.setPeriod_id(pinBO.getPeriod_id());

       int result = myPinDAO.insertPin(myPin);

       return result;
   }
```

Εικόνα 5.47 – Υλοποίηση της μεθόδου insertPin(PinsBOpinBO)

Η μέθοδος getPin() (εικόνα 5.48)παρέχει στον χρήστη-διαχειριστή την ικανότητα να ανακτήσει το αντικείμενο το οποίο εισήγαγε στην βάση δεδομένων ώστε να το δεί στο previewmode της εφαρμογής διαχείρισης. Λαμβάνει σαν όρισμα toid του Pinto οποίο εισαχθεί στην βάση δεδομένων και το επιστρέφει μέσω της μεθόδου getPin() (εικόνα 5.49η οποία ανακτά μέσω της μεθόδουpinDAO.getPinByID() το αντικείμενο PIN και στην συνέχεια για να επιστραφεί στον χρήστη μετατρέπεται σε PinBO.

```java
@GET
@Path("/getPin/{id}")
@Produces({MediaType.APPLICATION_JSON})
public String getPin(@PathParam("id") String pinId) throws IOException {

    PinsBO result = null;

    try {
        result = getPin(Integer.parseInt(pinId));


    } catch (Exception ex) {
        ex.printStackTrace();
    }
    return result.pinBoToJson(result);
```

Εικόνα 5.48 - Υλοποίηση της μεθόδου-service getPin()

```java
public PinsBO getPin(int pinId) {
    try {
        DAOFactory mySqlFactory = DAOFactory.getDAOFactory(DAOFactory.MYSQL);
        PinsDAO pinsDAO = mySqlFactory.getPinsDAO();

        PinsBO mypinBo = new PinsBO();

        Pins pin = pinsDAO.getPinById(pinId);

        mypinBo = pin.toPinsBO();

        return mypinBo;
    } catch (Exception e) {
        return null;
    }
}
```

Εικόνα 5.49- Υλοποίηση της μεθόδου getPin(intpinId)

Η μέθοδος-ServicegetNonPointPinsListJSON() (εικόνα 5.50)χρησιμοποιείται από τον χρήστη-διαχειριστή για την ανάκτηση όλων εκείνων των στιγμιότυπωνPinτα οποία δεν δείχνουν σε κάποιο μοντέλο. Χρησιμοποιείται για να έχει στην διάθεση του ο χρήστης-διαχειριστής την λίστα των Pinτα οποία έχει εισάγει στην βάση δεδομένων αλλά αν επιλεχτούν δεν οδηγούν πουθενά. Η λίστα αυτή ανακτάται μέσω της μεθόδου getNonPointPinsList() (εικόνα 5.51) η οποία καλεί την συνάρτηση pinsDAO.getNonPointingPeriodPins() για την ανάκτηση της λίστας από την βάση δεδομένων και εν συνεχεία την μετατρέπει σε λίστα αντικειμένων PinsBO. Εν συνεχεία μέσω της pinsListToJSON() γίνεται η μετατροπή της σε αντικείμενα JSON ώστε να καταλήξει στον χρήστη-διαχειριστή.

```java
@GET
    @Path("/getNonPointPinsList/{period}")
    @Produces({MediaType.APPLICATION_JSON})
    public String getNonPointPinsListJSON(@PathParam("period") int period ) throws IOException {

        String result = null;


        try {
            result = pinsListToJSON(getNonPointPinsList());

        } catch (Exception ex) {
            ex.printStackTrace();
        }
        return result;
    }
```

Εικόνα 5.50 - Υλοποίηση της μεθόδου-service getNonPointPinsListJSON ()

```
public List<PinsBO> getNonPointPinsList() {

        try {
            DAOFactory mySqlFactory = DAOFactory.getDAOFactory(DAOFactory.MYSQL);
            PinsDAO pinsDAO = mySqlFactory.getPinsDAO();

            List<PinsBO> pinsBO_list = new ArrayList<PinsBO>();

            List<Pins> model_list = pinsDAO.getNonPointingPeriodPins();

            for (int i = 0; i < model_list.size(); i++) {
                pinsBO_list.add(model_list.get(i).toPinsBO());
            }

            return pinsBO_list;
        } catch (Exception e) {
            return null;
        }
}
```

Εικόνα 5.51 – Υλοποίηση της μεθόδου getNonPointPinsList()

```
public String pinsListToJSON(List<PinsBO> pinsList) {

        String pinsListJson = "{ \"Pins\" :[\n";

        for (int i = 0; i < pinsList.size(); i++) {

            pinsListJson = pinsListJson+pinsList.get(i).pinBoToJson(pinsList.get(i));

            if (i != pinsList.size() - 1) {
                pinsListJson = pinsListJson + ",\n";
            }
        }
        pinsListJson = pinsListJson + "\n]}";
        return pinsListJson;
    }
```

Εικόνα 5.52 – Υλοποίηση της μεθόδου pinsListToJSON(List<PinsBO>pinsList)

5.4.3 Scene

Η κατηγορία Scene των υπηρεσιών που παρέχονται από την εφαρμογή μας, αφορά τον χρήστη-επισκέπτη. Υπάρχει μόνο μία μέθοδος τύπου get, καθώς δεν επιτρέπεται στον χρήστη-επισκέπτη κάποια άλλη λειτουργία πέραν της ανάκτησης δεδομένων για προεπισκόπηση. Η μέθοδος getSceneJSON(StringpointsId) λαμβάνει σαν όρισμα το πρωτεύον κλειδί του μοντέλου που θέλει ο χρήστης να εμφανιστεί στην οθόνη του, και καλεί την getSceneJson().

```java
@Path("/scene")
public class SceneRESTService {
      @GET
    @Path("/getScene/{id}")
    @Produces({MediaType.APPLICATION_JSON})
    public String getSceneJSON(@PathParam("id") String pointsId) throws IOException {

        String result = null;

        try {
            result = getSceneJson( Integer.parseInt( pointsId));

        } catch (Exception ex) {
            ex.printStackTrace();
        }
        return result;
    }
}
```

Εικόνα 5.53 – Υλοποίηση της κλάσης-serviceSceneRestService

```java
public String getSceneJson(int pointsId){
         String result=null;
        try {
            DAOFactory mySqlFactory = DAOFactory.getDAOFactory(DAOFactory.MYSQL);
            ModelDAO modelDAO = mySqlFactory.getModelDAO();
            PinsDAO pinsDAO = mySqlFactory.getPinsDAO();

            ModelBO groundBO = getModel(pointsId);

            List<Model> model_list = modelDAO.getModelListByGround(pointsId);

            List<ModelBO> modelBO_list = new ArrayList<ModelBO>();

            for (Model model_list1 : model_list) {
                modelBO_list.add(model_list1.toModelBO());
            }

            List<Pins> pins_list = pinsDAO.getPinsByGround(pointsId);

            List<PinsBO> pinsBO_list = new ArrayList<PinsBO>();

            for (Pins pins_list1 : pins_list) {
                pinsBO_list.add(pins_list1.toPinsBO());
            }

            result = boSceneToJson( groundBO,  modelBO_list,  pinsBO_list);

        } catch (Exception e) {
            return null;
        }
        return result;
    }
```

Εικόνα 5.54-Υλοποίηση της μεθόδου getSceneJSON()

Η μέθοδος getSceneJson(intpointsId) (εικόνα 5.54)λαμβάνει σαν όρισμα το πρωτεύον κλειδί του μοντέλου που θέλει ο χρήστης-επισκέπτης να ανακτήσει, και μέσω αυτού ανακτά το μοντέλο τύπου “GROUND” το οποίο είναι βασικός πυλώνας της σκηνής. Εν συνεχεία ανακτά με την μέθοδο modelDAO.getModelListByGround(pointsId) όλα τα αντικείμενα Model της βάσης δεδομένων τα οποία έχουν σαν foreignkey (groundId) το δοσμένο σαν όρισμα pointsId. Τέλος ανακτά όλα εκείνα τα pinsτα οποία ανήκουν και αυτά στο

συγκεκριμένο μοντέλο τύπου Ground. Μετατρέπουμε όλα τα ανακτηθέντα αντικείμενα σε BOαντικείμενα, και εν συνεχεία σε JSON μέσω της μεθόδου boSceneToJson(groundBO,modelBO_list,pinsBO_list) (εικόνα 5.55).

```
1  public String boSceneToJson(ModelBO groundBO, List<ModelBO> model_list, List<PinsBO> pins_list){
2
3          String result ;
4
5          String pinsListJson = " \"pins\" :[\n";
6
7          for (int i = 0; i < pins_list.size(); i++) {
8              pinsListJson = pinsListJson+pins_list.get(i).pinBoToJson(pins_list.get(i));
9
10             if (i != pins_list.size() - 1) {
11                 pinsListJson = pinsListJson + ",\n";
12             }
13         }
14
15         pinsListJson = pinsListJson + "\n]";
16
17         String modelListJson = " \"models\" :[\n";
18
19         for (int i = 0; i < model_list.size(); i++) {
20             modelListJson = modelListJson+model_list.get(i).modelBoToJson();
21
22             if (i != model_list.size() - 1) {
23                 modelListJson = modelListJson + ",\n";
24             }
25         }
26
27         modelListJson = modelListJson + "\n]";
28
29         String ground = groundBO.modelBoToJson();
30
31         result="{\"scene_to_import\":{\n"
32                 + pinsListJson +","
33                 + modelListJson+","
34                 + "\"ground\":"+ground
35                 +"}\n}";
36
37         return result;
38     }
```

Εικόνα5.55 – ΥλοποίησητηςμεθόδουboSceneToJson(ModelBO groundBO, List<ModelBO> model_list, List<PinsBO> pins_list)

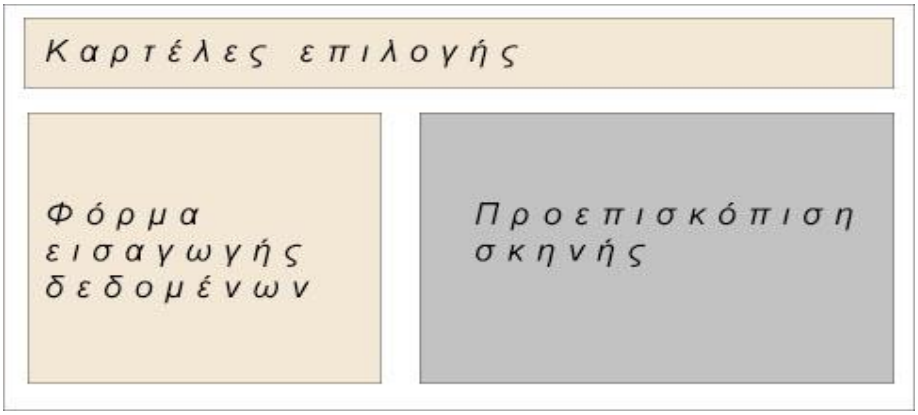
5.5 Γραφική Διεπαφή Χρήστη

Σε αυτή την ενότητα θα μελετήσουμε τον τρόπο με τον οποίο υλοποιήσαμε την γραφική διεπαφή του χρήστη. Θα διαχωρίσουμε τις δύο εφαρμογές, ανάλογα με τον χρήστη που αφορά η κάθε μία, σε εφαρμογή-διαχείρισης και εφαρμογή-επισκέπτη. Αυτό προκύπτει καθώς διαφορετικές απαιτήσεις είχαμε για την υλοποίηση της εφαρμογής-επισκέπτη όσων αφορά την γραφική διεπαφή, και διαφορετικές για την εφαρμογή-διαχείρισης.

5.5.1 Εφαρμογή διαχείρισης

Η εφαρμογή διαχείρισης αφορά τον διαχειριστή της εφαρμογής επισκέπτη, όπου για την συγκεκριμένη εφαρμογή τον ρόλο του διαχειριστή είχα προσωπικά. Αυτό μας

έδινε την ευελιξία να μην αναλωθούμε στην υλοποίηση ενός φιλικού περιβάλλοντος διαχείρισης καθώς βασικός στόχος της διπλωματικής ήταν η ανάπτυξη της εφαρμογής επισκέπτη η οποία έπρεπε να παραδοθεί στην CytaHellas. Παρόλα αυτά, προσπαθήσαμε να υλοποιήσουμε μία φιλική εφαρμογή διαχείρισης, η οποία με περεταίρω βελτιώσεις, και έναν οδηγό διαχείρισης να μπορεί να αναπτυχθεί. Ο αριθμός των τρισδιάστατων μοντέλων τα οποία υλοποίησε το τμήμα της αρχιτεκτονικής σχολής ήταν αρκετά μεγάλος για την εισαγωγή στην εφαρμογή μας, επομένως με το φιλικό αυτό σύστημα αποφύγαμε τα λάθη και πραγματοποιήσαμε πιο εύκολα την εισαγωγή τους.



Εικόνα 5.56 – Γενική διάταξη της γραφικής διεπαφής της εφαρμογής διαχείρισης.

Για την υλοποίηση της γραφικής διεπαφής της εφαρμογής διαχείρισης, χρησιμοποιήσαμε οκτώ καρτέλες οι οποίες αντιστοιχούν στις εφτά εποχές που καλύπτει η εφαρμογή μας, και μία καρτέλα η οποία αφορά το τύπου “GROUND” τρισδιάστατο αντικείμενο. Κάθε καρτέλα περιέχει μία φόρμα εισαγωγής δεδομένων, ενώ ενιαίο είναι για κάθε καρτέλα ένα στοιχείο “canvas” στο οποίο γίνεται η προεπισκόπηση της σκηνής που δημιουργείται κατά την εισαγωγή κάθε μοντέλου.

Για την υλοποίηση των καρτελών χρησιμοποιήσαμε ένα HTMLelement<ul>, δηλαδή μία λίστα όπου κάθε στοιχείο της αφορά κάθε μία από τις καρτέλες που δημιουργήσαμε (εικόνα 5.57). Χρησιμοποιώντας τον κατάλληλο κώδικα μορφοποίησης CSSκαθώς και συναρτήσεις Javascript, υλοποιήσαμε την εναλλαγή περιεχομένου κάθε καρτέλας. Όλες οι καρτέλες περιέχουν την ίδια φόρμα, καθώς κάθε μοντέλο περιγράφεται από τα ίδια δεδομένα χωρίς να έχει σημασία ποιάς εποχής είναι, αλλά με την μέθοδο καρτελών ενισχύσαμε τον έλεγχο του διαχειριστή για την σωστή εισαγωγή των τρισδιάστατων μοντέλων. Αυτό επιτεύχθηκε καθώς έχουμε ένα στοιχείο canvas στο οποίο εμφανίζεται η σκηνή μας, και τα μοντέλα που εισάγονται, αλλά εναλλάσσοντας τις καρτέλες εποχής, εναλλάσσεται και η σκηνή προσφέροντας την διαδραστικότητα που θα βλέπει ο χρήστης-πελάτης στην εφαρμογή.

```
<ul class="nav nav-tabs" role="tablist" id="myTab">
    <li><a href="#ground">Ground</a></li>
    <li><a href="#modern">Modern</a></li>
    <li><a href="#ottoman">Ottoman</a></li>
    <li><a href="#venetian">Venetian</a></li>
    <li><a href="#byzantine">Byzantine</a></li>
    <li><a href="#roman">Roman</a></li>
    <li><a href="#hellinistic">Hellinistic</a></li>
    <li><a href="#minoan">Minoan</a></li>
</ul>
```

Εικόνα 5.57 – Κώδικας HTML: Καρτέλες επιλογής

Για την υλοποίηση της φόρμας ground (εικόνα 5.58), χρησιμοποιήσαμε το HTMLelement “form” η οποία περιέχει τα εξής δεδομένα για εισαγωγή: Name, obj_file, mtl_file, Level, Clickable, Hover, Scale, ComeFromPin.

```
<form role="form" name="ground_form" >
    <div class="form-group">
        <label for="input_name">Model Name</label>
        <input type="text" class="form-control" name="model_name" id="input_name">
    </div>
    <div class="form-group">
        <label for="obj_file">OBJ file input</label>
        <input type="file" name="obj" id="obj_file" >
    </div>
    <div class="form-group">
        <label for="mtl_input">MTL File input</label>
        <input type="file" name="mtl" id="mtl_input">
    </div>
    <div class="form-group" id="level_check">
        <label for="inputLvl">Level</label>
        <select name="level" class="form-control">
            <option selected="selected" value="null">Not selected Yet</option>
        </select>
    </div>
    <div class="form-group">
        <label for="">Clickable</label>
        <select name="clickable" class="form-control">
            <option selected="selected" value="FALSE">No</option>
            <option  value="TRUE">Yes</option>
        </select>
    </div>
    <div class="form-group">
        <label for="">Hover</label>
        <select name="hover" class="form-control">
            <option selected="selected" value="FALSE">No</option>
            <option  value="TRUE">Yes</option>
        </select>
    </div>
    <div class="form-group">
        <label for="">Scale</label>
        <input  type="range" id="range_sl" min="0.001" max="100" value="1" step="0.001" />
            <br>
        <input name="scale1" type="text" id="rangevalue" value="1">
    </div>
    <div class="form-group">
        <label for="">Come from Pin:</label>
        <select name="pin_points_to" class="form-control">
            <option selected="selected" value="">Not selected Yet</option>
        </select>
    </div>
    <input id="obj_url_db" type="hidden" name="obj1" >
    <input id="mtl_url_db" type="hidden" name="mtl1" >
    <input type="hidden" name="type" value="GROUND" >
    <input type="hidden" name="ground_id" value="0" >

</form>
```

Εικόνα 5.58 – Υλοποίηση φόρμας, καρτέλα GROUND

Οι υπόλοιπες φόρμες, έχουν μερικά από τα στοιχεία που περιλαμβάνονται στην φόρμα Ground, καθώς χρησιμοποιούμε το groundσαν βάση της σκηνής. Τα στοιχεία που περιέχουν οι υπόλοιπες φόρμες(ανά εποχή) είναι τα εξής: Name, obj_file, mtl_file, Clickable, Hover, Scale.

Για την αποστολή των δεδομένων της εκάστοτε φόρμας στην βάση δεδομένων, καλείται μέσω της επιλογής του κουμπιού «InsertModel», η απαραίτητη συνάρτηση «PostForm()» η οποία μέσω της μεθόδου AJAX επικοινωνεί με την πλευρά του διακομιστή (εικόνα 5.59). Η εκάστοτε φόρμα περνάει σαν όρισμα στην συνάρτηση PostForm, και μετατρέπεται μέσω της συνάρτησης modelFormToJSON(form) σε

αντικείμενο τύπου JSON, καθώς χρησιμοποιούμε την μέθοδο AJAX με τύπο δεδομένων JSON.Επιπλέον στον παρακάτω κώδικα βλέπουμε ότι με την ολοκλήρωση εισαγωγής του μοντέλου στη βάση δεδομένων ανακτούμε το αντικείμενο αυτό και καλείται η συνάρτηση getAndPreview() η οποία στο σώμα της καλεί την συνάρτηση με την οποία γίνεται η προεπισκόπηση του τρισδιάστατου αντικειμένου στο canvaselement. Με τον τρόπο με τον οποίο τα δεδομένα αυτά τα αναπαριστούμε στο canvas θα ασχοληθούμε σε επόμενη ενότητα.

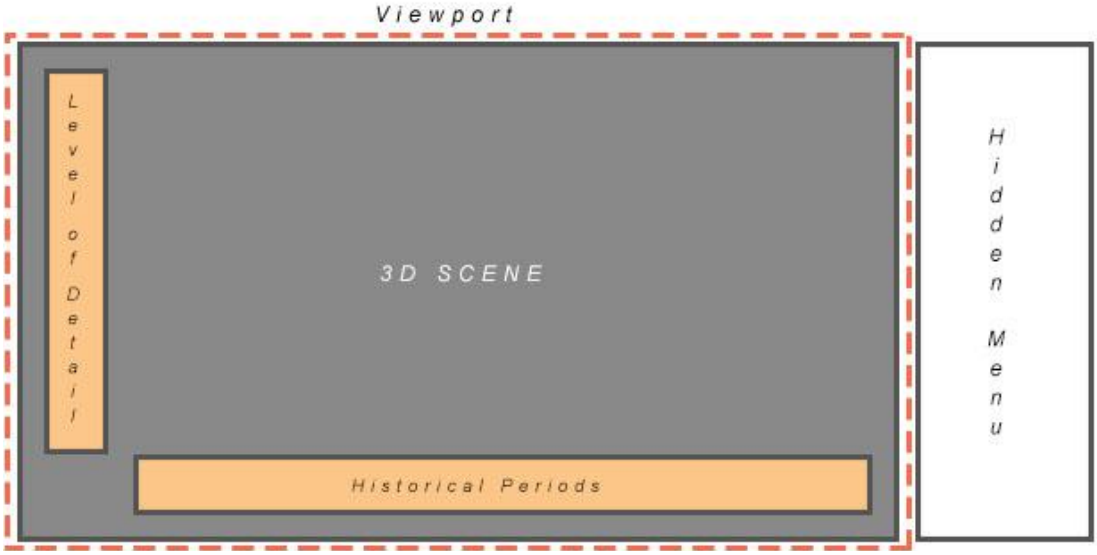
```
function PostForm(form, period) {

    var mymodel_json = modelFormToJSON(form);
    if (form == "ground_form"){
        mymodel_json.comeFromPin= document.forms['ground_form']['pin_points_to'].value;
    }
    var xmlHttp = new XMLHttpRequest();
    xmlHttp.open("POST", "http://localhost:8080/Crete3D/rest/model", true);
    xmlHttp.setRequestHeader('Content-Type', 'application/json');
    xmlHttp.onreadystatechange = function () {
        if (xmlHttp.readyState == 4) {
            if (xmlHttp.status == 200) {
                console.log('Server Response : ' + xmlHttp.responseText);
                selected = JSON.parse(xmlHttp.responseText);
                if (form == "ground_form") {
                    my_ground_json = selected;
                    my_ground_id = selected.id;
                    model_type = "ground";
                } else {
                    model_type = period;

                }
                getAndPreview(selected.id, model_type);


            } else if (xmlHttp.status == 500) {
                console.log('Server ERROR response : ' + xmlHttp.responseText);
            }
        }
    };
    xmlHttp.send(JSON.stringify(mymodel_json));
}
```

Εικόνα 5.59 – Υλοποίηση της εισαγωγής τρισδιάστατου μοντέλου μέσω AJAX

Με τον ίδιο τρόπο εισάγονται τα Pins τα οποία θέλει ο χρήστης διαχειριστής να εισάγει, τα οποία όμως έχουν ένα επιπλέον στοιχείο το οποίο είναι η επιλογή εποχών, δηλαδή σε ποιές από τις εφτά εποχές θα εμφανίζεται το κάθε pin.

5.5.2 Εφαρμογή πελάτη

Η εφαρμογή-πελάτη αφορά τον χρήστη-πελάτη, και αφορά την βασική εφαρμογή της διπλωματικής μας. Μέσω της εφαρμογής διαχείρισης εισάγουμε όλα τα τρισδιάστατα αντικείμενα εκείνα τα οποία θέλουμε να εμφανιστούν στην εφαρμογή-πελάτη. Για την γραφική διεπαφή της εφαρμογής αυτής, μας δόθηκαν σαφείς οδηγίες καθώς και εικόνες σχεδιασμένες στο Photoshop, οι οποίες αποτελούσαν το προσδοκώμενο γραφικό αποτέλεσμα το οποίο μας ζητήθηκε. Ως εκ τούτου προσπαθήσαμε να αναπαραστήσουμε με κώδικα HTML,Javascript και CSSτο αποτέλεσμα το οποίο μας ζητήθηκε.



Εικόνα 5.60 - Γενική διάταξη της γραφικής διεπαφής της εφαρμογής πελάτη.

Βασικός πυλώνας της εφαρμογής πελάτη καθώς και της διπλωματικής αυτής είναι η απεικόνιση των τρισδιάστατων μοντέλων στη 3Dσκηνή. Η σκηνή αυτή υλοποιείται σε κώδικα HTML μέσω του elementcanvas. Στο canvasμέσω κώδικα javascriptκαι με την χρήση της βιβλιοθήκης Three.jsαπεικονίζεται κάθε τρισδιάστατη σκηνή η οποία ανακτάται από την βάση δεδομένων.

Κατά την διάρκεια φόρτωσης της σελίδας καλείται μέσω της συνάρτησηςonload() (εικόνα 98),η οποία εκτελείται την στιγμή την οποία φορτώνεται η σελίδα που έχουμε πληκτρολογήσει στον φυλλομετρητή, καλούνται με την σειρά οι εξής συναρτήσεις:

- initVariables()
- init()
- animate()
- getAndPreviewSceneFromDb(creteId)

```
1 <script>
2     window.onload(initVariables(), init(), animate(), getAndPreviewSceneFromDb(creteId));
3 </script>
```

Εικόνα 5.61 – Κλήση των συναρτήσεων Javascriptμέσω της μεθόδου window.onload().

Θα αναλύσουμε την λειτουργία της κάθε συνάρτησης που καλείται παραπάνω ώστε να εξηγηθεί ο τρόπος με τον οποίο εμφανίζονται τα τρισδιάστατα μοντέλα στον φυλλομετρητή του χρήστη-πελάτη.

Η συνάρτηση initVariables() καλείται για την αρχικοποίηση των μεταβλητών οι οποίες είναι απαραίτητες να χρησιμοποιηθούν για την υλοποίηση. Πιο συγκεκριμένα σε αυτή τη συνάρτηση ορίζουμε το javascriptαντικείμενο το οποίο χρησιμοποιείται για την αναπαράσταση της τρισδιάστατης σκηνής και το ονομάσαμε my_scene. Επιπλέον αρχικοποιούμε δύο ακόμη μεταβλητές, αντικείμενα της javascript, οι οποίες είναι οι TargetListκαι η HoverList, οι οποίες θα χρησιμοποιηθούν με σκοπό να κρατάμε στην μία -TargetList- τα τρισδιάστατα αντικείμενα εκείνα τα οποία μπορεί να επιλέξει ο χρήστης, δηλαδή να αλληλοεπιδράσει μαζί τους, καθώς και στην HoverListτα τρισδιάστατα αντικείμενα τα οποία δίνουν στον χρήστη πληροφορία περνώντας τον κέρσορα από πάνω τους. Το javascriptobjectmy_scene αποτελείται από επιμέρους αντικείμενα τα οποία ορίζουν ουσιαστικά την σκηνή τα οποία είναι τα pins, το ground και το models, το οποίο εμπεριέχει εφτά αντικείμενα τα οποία

αφορούν τις εποχές. Κάθε ένα από τα αντικείμενα (models, pins, ground) περιέχουν δύο λίστες εκ των οποίων η μία -json- κρατάει τα τρισδιάστατα μοντέλα που έχουν ληφθεί από τον serverκαι η άλλη –loaded3d- τα τρισδιάστατα αντικείμενα που έχουν ληφθεί από τον server αλλά έχουν φορτωθεί και στην σκηνή μας. Ορίσαμε λοιπόν μέσω της συνάρτησης InitVariables() την δομή που θέλουμε να έχει η μεταβλητή η οποία θα «κρατάει» την σκηνή που εμφανίζεται στον φυλλομετρητή.

```
function initVariables() {
    my_scene = {
        pins: {
            json: [],
            loaded3d: []
        },
        ground: {
            json: [],
            loaded3d: []
        },
        models: {
            modern: {
                json: [],
                loaded3d: []
            },
            ottoman: {
                json: [],
                loaded3d: []
            },
            venetian: {
                json: [],
                loaded3d: []
            },
            byzantine: {
                json: [],
                loaded3d: []
            },
            roman: {
                json: [],
                loaded3d: []
            },
            hellinistic: {
                json: [],
                loaded3d: []
            },
            minoan: {
                json: [],
                loaded3d: []
            }
        }
    };

    targetList = [];
    hoverList = [];
}
```

Εικόνα 5.62- Υλοποίηση της συνάρτησης initVariables()

Στην συνέχεια καλείται η init() η οποία αρχικοποιεί όλα όσα συνθέτουν μία σκηνή για τρισδιάστατη απεικόνιση στον φυλλομετρητή. Σε αυτό το σημείο αρχίζει και η

χρήση της βιβλιοθήκης Three.jsμέσω της οποίας χρησιμοποιείται η webglόπως εξηγήσαμε σε προηγούμενα κεφάλαια.

```
function init() {
    // SCENE
    scene = new THREE.Scene();
    scene.fog = new THREE.Fog(0xffffff, 1000, 10000);

    // CAMERA
    SCREEN_WIDTH = window.innerWidth, SCREEN_HEIGHT = window.innerHeight;
    var VIEW_ANGLE = 45, ASPECT = SCREEN_WIDTH / SCREEN_HEIGHT, NEAR = 0.1, FAR = 20000;
    camera = new THREE.PerspectiveCamera(VIEW_ANGLE, ASPECT, NEAR, FAR);
    camera.position.x = 0;
    camera.position.y = 100;
    camera.position.z = 100;
    scene.add(camera);

    // RENDERER
    if (Detector.webgl) {
        renderer = new THREE.WebGLRenderer({antialias: true});
    } else {
        renderer = new THREE.CanvasRenderer();
    }
    renderer.setSize(SCREEN_WIDTH, SCREEN_HEIGHT);
    renderer.setClearColor(0x333333, 1);

    container = document.getElementById('ThreeJS');
    container.appendChild(renderer.domElement);
```

Εικόνα 5.63 – Υλοποίηση της συνάρτησης init() – Scene,Camera,Renderer

Όπως βλέπουμε στο πρώτο μέρος της υλοποίησης της συνάρτησης init() (εικόνα 5.63), ορίζουμε τα τρία βασικά στοιχεία που χρειάζονται για την απεικόνιση της τρισδιάστατης σκηνής μας στον φυλλομετρητή. Αρχικά ορίζουμε την “σκηνή”, όπου για την υλοποίηση της εφαρμογής μας κάνουμε χρήση της Three.scenεη οποία παρέχεται από την βιβλιοθήκη Three.js. Στην συνέχεια, αφού έχουμε ορίσει την σκηνή ορίζουμε την camera, θέτοντας το οπτικό πεδίο στις 45 μοίρες και την αναλογία όσο η αναλογία του ύψους του παραθύρου του φυλλομετρητή προς το πλάτος του. Ακόμη ορίσαμε την θέση της κάμερας και την εισήγαμε στην σκηνή. Στην συνέχεια ορίσαμε το “μέσο” το οποίο μας προσφέρει την τρισδιάστατη απεικόνιση στον φυλλομετρητή. Αυτό είναι ο WebGLRendererαν υποστηρίζεται από το hardwaretου χρήστη, ειδάλλως το CanvasRenderer, το οποίο όμως δεν έχει την ίδια απόδοση με τον WebGLRenderer. Αφού ορίσουμε τον renderer, επιλέγουμε το στοιχείο στο οποίο θα εισαχθεί, μέσω της μεθόδου appendChild, το οποίο έχουμε ορίσει στον κώδικα HTML ώς ένα στοιχείο div, με id=”ThreeJS”.

Στην συνέχεια ορίσαμε τον τρόπο αλληλεπίδρασης του χρήστη με την τρισδιάστατη σκηνή. Χρησιμοποιήσαμε το THREE.OrbitControls() (εικόνα 5.64) της βιβλιοθήκης three.jsτα οποία ουσιαστικά παρέχουν μία ομαλή χρήση μετακίνησης της κάμερας, τοποθετώντας νοητά την τρισδιάστατη σκηνή πάνω σε μία σφαίρα, και «τραβώντας» με το ποντίκι την σκηνή, ουσιαστικά περιστρέφεται η σφαίρα πάνω στην οποία πατάει η σκηνή. Χρησιμοποιήσαμε σε αυτό το σημείο τον πρώτο EventListener, ο οποίος κάθε φορά που χρησιμοποιούνται τα OrbitControls, καλεί την συνάρτηση render() την οποία θα εξετάσουμε παρακάτω.

```

    // CONTROLS
    controls = new THREE.OrbitControls(camera);
    controls.damping = 0.2;
    controls.addEventListener('change', render);
```

Εικόνα 5.64 - Υλοποίηση της συνάρτησης init() – Controls

Έχοντας ορίσει τα βασικά στοιχεία για να μπορεί να δημιουργηθεί μία τρισδιάστατη σκηνή, μέσω της init() ορίζουμε και τον φωτισμό της σκηνής έτσι ώστε να μπορούν να φαίνονται τα τρισδιάστατα αντικείμενα και να μην είναι μαύρα, όπως θα ήταν και στην πραγματικότητα μέσα σε ένα αποκλεισμένο από οποιαδήποτε πηγή φωτός δωμάτιο.

```
32      //LIGHTS
33
34      var directionalLight1 = new THREE.DirectionalLight(0xffffff);
35      directionalLight1.position.x = -180;
36      directionalLight1.position.y = 70;
37      directionalLight1.position.z = -180;
38      directionalLight1.castShadow = false;
39      directionalLight1.intensity = 0.90;
40      scene.add(directionalLight1);
41
42      var directionalLight = new THREE.DirectionalLight(0xffffff);
43      directionalLight.position.x = 480;
44      directionalLight.position.y = 470;
45      directionalLight.position.z = 280;
46      directionalLight.shadowBias = 0.0001;
47      directionalLight.castShadow = true;
48      directionalLight.shadowDarkness = 0.4;
49      directionalLight.intensity = 1.50;
50      directionalLight.shadowCameraNear = 1;
51      directionalLight.shadowCameraFar = 1000;
52      directionalLight.shadowCameraRight = 200;
53      directionalLight.shadowCameraLeft = -200;
54      directionalLight.shadowCameraTop = 200;
55      directionalLight.shadowCameraBottom = -200;
56      directionalLight.shadowMapWidth = 2048;
57      directionalLight.shadowMapHeight = 2048;
58      scene.add(directionalLight);
```

Εικόνα 5.65 - Υλοποίηση της συνάρτησης init() – Lights

Για την υλοποίηση της εφαρμογής μας ορίσαμε δύο πηγές φωτός τύπου DirectionalLight όπως φαίνεται και στον κώδικα (εικόνα 5.65), οι οποίες στον πραγματικό κόσμο αναπαριστώνται με την χρήση ενός φακού. Η χρήση των δύο συγκεκριμένων πηγών έγινε μετά από πολλές δοκιμές μέσα από πολλές επιλογές φωτισμού. Στην εφαρμογή μας η μία πηγή παίζει τον ρόλο του ήλιου, με αυξημένη ένταση, λευκό χρώμα και τοποθετημένη σε συγκεκριμένο σημείο μετά από δοκιμές, μέσω της οποίας παράγονται οι απαραίτητες σκιάσεις. Όμως αν είχαμε μόνο αυτή την πηγή φωτισμού, τότε ορισμένες επιφάνειες θα φαίνονταν μαύρες καθώς δεν διοχετεύεται φως προς αυτές μέσω άλλων αντικειμένων. Για τον λόγω αυτό τοποθετήθηκε μία δεύτερη πηγή φωτός η οποία έχει πολύ μικρότερη ένταση, έτσι ώστε να μην εξασθενούνται οι σκιές που προκαλεί η πρώτη, αλλά να φωτίζονται κατάλληλα οι σκοτεινές επιφάνειες.

Στην συνέχεια ορίζουμε τον projector, ο οποίος είναι απαραίτητος για να προβάλει ουσιαστικά τα υπολογισμένα μέσω της κάρτας γραφικών τρισδιάστατα αντικείμενα στην δισδιάστατη οθόνη του χρήστη. Τέλος για την ολοκλήρωση της υλοποίησης της init() ορίζουμε τους eventListenerstους οποίους χρειαζόμαστε κάθε φορά που ο χρήστης κάνει «κλικ» με το ποντίκι του, και κάθε φορά που κινεί τον κέρσορα, καθώς και όταν ο χρήστης αλλάζει το μέγεθος του παραθύρου του φυλλομετρητή, έτσι ώστε

να αλλάζει ανάλογα και το μέγεθος του Canvas καθώς και της τρισδιάστατης σκηνής. Θα αναφερθούμε στην συνέχεια στις συναρτήσεις οι οποίες καλούνται μέσω των EventListeners.

```
60      projector = new THREE.Projector();
61
62      document.addEventListener('mousedown', onDocumentMouseDown, false);
63      document.addEventListener('mousemove', onDocumentMouseMove, false);
64      document.addEventListener('mousemove', update, false);
65
66      window.addEventListener('resize', onWindowResize, false);
67
68  }
```

Εικόνα 5.66 - Υλοποίηση της συνάρτησης init() – Projector,EventListeners

Έχοντας υλοποιήσει λοιπόν την συνάρτηση init(), η επόμενη συνάρτηση η οποία καλείται όταν μέσω της συνάρτησης window.onload() είναι η animate(). Η animate() καλείται στις περισσότερες εφαρμογές με χρήση της WebGlέτσι ώστε να καλείται εξήντα φορές ανά δευτερόλεπτο (60FPS) η συνάρτηση render() η οποία είναι αυτή που ζωγραφίζει πάνω στο canvasτην σκηνή. Χρησιμοποιώντας την αυτούσια παρατηρήσαμε ότι καταναλώναμε πολλούς πόρους από την υπολογιστική δύναμη με αποτέλεσμα η εφαρμογή να μην είναι όσο αποδοτική έπρεπε σε έναν μέσο υπολογιστή. Για τον λόγο αυτό, τροποποιήσαμε την βιβλιοθήκη Three.jsέτσι ώστε να καλείται η συνάρτηση render() μέσω της animate() μόνο όσες φορές χρειάζεται μέχρις ώτου να φορτωθούν τα μοντέλα και να «ζωγραφιστούν» στην σκηνή. Επιπλέον την καλούμε κάθε φορά όπου ο χρήστης αλληλοεπιδρά με την σκηνή, είτε περιστρέφοντας την, είτε μεταβαίνοντας από μία σκηνή σε μία άλλη. Αυξήσαμε λοιπόν την απόδοση της εφαρμογής μας, καταναλώνοντας μόνο όσους πόρους είναι απαραίτητοι, και όχι σπαταλώντας τους όση ώρα ο χρήστης βρίσκεται στην εφαρμογή ακόμη και αν δεν αλληλοεπιδρά με αυτή. Η animate() τρέχει επαναληπτικάκαλώντας την συνάρτηση render() μέχρις ότου να έχει ολοκληρωθεί η αλληλεπίδραση του χρήστη με την σκηνή (εικόνα 5.67).

```
1   function animate()
2   {
3       id = requestAnimationFrame(animate);
4
5       if (animActive && render_complete) {
6           clcAnim();
7       }
8
9       render();
10  }
```

Εικόνα 5.67 – Υλοποίηση της συνάρτησης animate()

Μέχρι αυτό το σημείο έχουμε δημιουργήσει ότι χρειάζεται ώστε να αρχίσουμε να τοποθετούμε στην σκηνή τα τρισδιάστατα αντικείμενα τα οποία θέλουμε. Για τον λόγω αυτό καλείται η τελευταία συνάρτηση μέσω της onload() η οποία είναι η getAndPreviewSceneFromDb() (εικόνα 5.68). Ησυνάρτηση αυτή καλεί μέσω AJAX toRestfulServicegetScenetης κατηγορίας sceneπου αναλύσαμε προηγουμένως, ώστε να γίνει η ανάκτηση της πρώτης σκηνής η οποία εμφανίζεται στην εφαρμογή μας. Επίσης η ανάκτηση κάθε σκηνής γίνεται μέσω αυτή της συνάρτησης ανάλογα με το όρισμα το οποίο περνάει κάθε φορά που καλείται. Αποστέλλεται λοιπόν το αίτημα

στον server, και μόλις επιστραφεί η απάντηση ασύγχρονα, καλείται η συνάρτηση loadScene(response). Να θυμίσουμε ότι η μέθοδος getSceneεπιστρέφει το JSONαντικείμενο το οποίο περιέχει όλα τα στοιχεία της σκηνής που προβάλεται που δημιουργήσαμε στον server.

```
function getAndPreviewSceneFromDb(id) {
    var hostName = window.location.host;
    var xmlHttp = new XMLHttpRequest();
    xmlHttp.open("GET", "http://" + hostName + "/Crete3D/rest/Scene/getScene/" + id, true);
    xmlHttp.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
    xmlHttp.onreadystatechange = function () {
        if (xmlHttp.readyState == 4) {
            if (xmlHttp.status == 200) {
                var response = xmlHttp.responseText;
                response = JSON.parse(response);
                loadScene(response);

            }
        }
    };
    xmlHttp.send();
}
```

Εικόνα 5.68 – Υλοποίηση της συνάρτησης getAndPreviewSceneFromDB()

Μέσω της loadScene() διαχωρίζουμε τα αντικείμενα pins, groundκαι model, και καλείται η συνάρτηση loadSceneLoader() για το κάθε ένα από αυτά. Η συνάρτηση αυτή είναι υπεύθυνη για την απεικόνιση του τρισδιάστατου αντικειμένου το οποίο λαμβάνει σαν όρισμα μέσω του αρχείου objκαι mtl. Φορτώνει ασύγχρονα το αντικείμενο μέσω των αρχείων και ουσιαστικά το μεταφράζει έτσι ώστε να είναι έτοιμο για απεικόνιση στη σκηνή όταν το χρειαστούμε. Ορίζεται το μέγεθος του, σύμφωνα με την τιμή scale, δίνεται όνομα στο τρισδιάστατο αντικείμενο, ούτως ώστε να χρησιμοποιηθεί σε tooltip. Επιπλέον μέσω αυτής της συνάρτησης εισάγουμε στην μεταβλητή my_sceneτο κάθε αντικείμενο, καθώς έχει φορτωθεί αλλά θέλουμε να μπορέσουμε να το καλέσουμε οποτεδήποτε χρειαστεί ανάλογα με την αλληλεπίδραση του χρήστη. Συγκεντρωτικά, φορτώνουμε το κάθε αντικείμενο της σκηνής που ανακτήθηκε, αλλά δεν το περνάμε αμέσως στην σκηνή που βλέπει ο χρήστης. Αυτό συμβαίνει διότι για να αποφύγουμε την καθυστέρηση της επικοινωνίας με τον server σε μετάβαση από εποχή σε εποχή, ανακτάμε όλες τις εποχές, επομένως δεν θα μπορούσαμε να εμφανίζουμε όλα τα μοντέλα την ίδια στιγμή. Τα φορτώνουμε και τα κρατάμε στην μεταβλητή αντικείμενο my_scene, και καλούμε για απεικόνιση το κατάλληλο κάθε φορά. Όπως είπαμε και παραπάνω το «φόρτωμα» των τρισδιάστατων αντικειμένων γίνεται ασύγχρονα επομένως δεν μπορούμε στο τέλος αυτής της συνάρτησης να καλέσουμε την συνάρτηση η οποία είναι υπεύθυνη για την απεικόνιση των σωστών τρισδιάστατων μοντέλων ανάλογα με την εποχή. Για αυτό υλοποιήσαμε έναν LoaderManager, ο οποίος μετράει πόσα αντικείμενα έχουν αποσταλεί για φόρτωμα, και πόσα είναι έτοιμα, επομένως όταν ολοκληρωθούν όλα, τότε καλούμε την συνάρτηση setSceneByPeriod(period). Τοόρισμα period, παίρνει τιμή από τα htmlκουμπιά, ανάλογα με το πιο έχει επιλεχτεί κάθε φορά. Στην συνάρτηση setSceneByPeriod() ανάλογα με την επιλεγμένη περίοδο, επιλέγονται ποιά από τα τρισδιάστατα μοντέλα θα εμφανιστούν στην σκηνή μέσω της εντολής scene.add(object) όπου objectείναι το κάθε τρισδιάστατο μοντέλο το οποίο υπάρχει στην μεταβλητή-αντικείμενο my_scene.

Μέχρι αυτό το σημείο προβάλλονται τα τρισδιάστατα μοντέλα που συγκροτούν μία σκηνή ανάλογα με την εποχή της και το επίπεδο λεπτομέρειας. Όμως ακόμη η μόνη διαδραστικότητα που έχει ο χρήστης με την σκηνή είναι η περιστροφή του αντικειμένου καθώς και η χρήση του zoom-in και zoom-outμέσω της ροδέλας του ποντικιού. Η λογική της εφαρμογής για την αλληλεπίδραση της εφαρμογής με τον χρήστη ούτως ώστε να βλέπει μία ταμπέλα-tooltipπερνώντας τον κέρσορα πάνω από

ένα pinκαθώς και η ανάκτηση της αντίστοιχης σκηνής αν επιλεχτεί το pinπραγματοποιείται μέσω των eventListeners, κίνησης του κέρσορα και κλίκ του ποντικιού,οι οποίοι καλούν συγκεκριμένες συναρτήσεις.

Η συνάρτηση onMouseMove() η οποία καλείται όταν το ποντίκι κινείται μέσα στην εφαρμογή που υλοποιούμε, κρατάει την θέσει του ποντικιού σε άξονες x,y στην οθόνη και καλεί την συνάρτηση update() (εικόνα 5.69).

```
function update()
{
    var vector = new THREE.Vector3(mouse.x, mouse.y, 1);
    vector.unproject(camera);
    var ray = new THREE.Raycaster(camera.position, vector.sub(camera.position).normalize());

    var intersects = ray.intersectObjects(targetList, true);

    if (intersects.length > 0)
    {
        if (intersects[ 0 ].object.clickable == "TRUE") {
            container.style.cursor = "pointer";
        }
        if (intersects[ 0 ].object != INTERSECTED)
        {
            if (INTERSECTED)
                INTERSECTED.material.color.setHex(INTERSECTED.currentHex);
            INTERSECTED = intersects[ 0 ].object;
            INTERSECTED.currentHex = INTERSECTED.material.color.getHex();
            INTERSECTED.material.color.setHex(0xFF6633);

            if (intersects[ 0 ].object.name)
            {
                if (!animActive) {
                    animate();
                }
                document.getElementById('tooltipTest').innerHTML = intersects[ 0 ].object.name;
                $('#tooltipTest').addClass("toolTipClass");
                $('#tooltipTest').css( "display", 'inline-block' );

            }

        }
    }
    else
    {
        container.style.cursor = "default";
        $("#tooltipTest").empty();
        $('#tooltipTest').css( "display", 'none' );
        if (INTERSECTED) {
            INTERSECTED.material.color.setHex(INTERSECTED.currentHex);
            if (!animActive) {
                animate();
            }
        }
        INTERSECTED = null;

    }
}
```

Εικόνα 5.69 – Υλοποίηση συνάρτησης update

Η συνάρτηση update() με λίγα λόγια είναι εκείνη η οποία ευθύνεται για την αλλαγή χρώματος κάθε τρισδιάστατου μοντέλου το οποίο μπορεί να επιλεχτεί από τον χρήστη, καθώς και για την εμφάνιση του tooltipτο οποίο εμφανίζει το όνομα του μοντέλου. Πιο συγκεκριμένα αν η νοητή προβολή του σημείου που βρίσκεται ο κέρσορας του ποντικιού προς την οθόνη προσπίπτει πάνω σε ένα αντικείμενο από τα οποία επιτρέπεται η αλληλεπίδραση με τον χρήστη, τότε αλλάζουμε το χρώμα του αντικειμένου ώστε να είναι διακριτό στον χρήστη, και εμφανίζουμε το tooltipσε θέση δίπλα στο ποντίκι, με το όνομα του αντικειμένου. Για την αλλαγή χρώματος, πρέπει να κληθεί η animate() καθώς πρέπει να ζωγραφιστεί πάλι η σκηνή αλλά με

διαφορετικό το χρώμα του αντικειμένου το οποίο δείχνει ο κέρσορας του ποντικιού. Αν η προβολή δεν προσπίπτει σε τέτοιο αντικείμενο, τότε αναιρείται η αλλαγή χρώματος που είχε γίνει στο προηγούμενο αντικείμενο, και εξαφανίζουμε το tooltip. Τα αντικείμενα τα οποία θέλουμε να αλληλοεπιδρούν με το ποντίκι ούτως ώστε όταν περνάει από πάνω τους ο κέρσορας να εμφανίζεται το tooltip, τα εισάγουμε στην λίστα που είχαμε ορίσει στην αρχή της κλήσης της εφαρμογής, targetList.

Για να πραγματοποιηθεί η μετάβαση από μία τρισδιάστατη σκηνή πρέπει να επιλεχθεί από τον χρήστη κάποιο από τα αντικείμενα τα οποία μπορούν να επιλεχτούν. Η συνάρτηση onDocumentMouseDownκαλείται όταν γίνει κλικ στην εφαρμογή και μέσω αυτής εντοπίζεται αν πρέπει να γίνει μετάβαση σε άλλη σκηνή.

```
function onDocumentMouseDown(event)
{

    // update the mouse variable
    mouse.x = (event.clientX / window.innerWidth) * 2 - 1;
    mouse.y = -(event.clientY / window.innerHeight) * 2 + 1;

    var vector = new THREE.Vector3(mouse.x, mouse.y, 1);
    vector.unproject(camera);
    ray = new THREE.Raycaster(camera.position, vector.sub(camera.position).normalize());

    var intersects = ray.intersectObjects(targetList, true);

    if (intersects.length > 0)
    {
        if (intersects[ 0 ].object.clickable == "TRUE") {
            if (my_scene.ground.json[0].levelId == "1") {
                press_left('perfecture');
                epipedo = 'perfecture';
            } else if (my_scene.ground.json[0].levelId == "2") {
                previous_prefecture_id = my_scene.ground.json[0].id;
                press_left('region');
                epipedo = 'region';
            } else if (my_scene.ground.json[0].levelId == "3") {
                previous_region_id = my_scene.ground.json[0].id;
                press_left('complex');
                epipedo = 'complex';
            } else if (my_scene.ground.json[0].levelId == "4") {
                previous_complex_id = my_scene.ground.json[0].id;
                press_left('monument');
                epipedo = 'monument';
            } else if (my_scene.ground.json[0].levelId == "5") {
                previous_monument_id = my_scene.ground.json[0].id;
            }
            clearScene();
            clearTargetList();
            clearSceneModels();
            clearSceneGround();
            clearScenePins();
            }


            getAndPreviewSceneFromDb(intersects[ 0 ].object.sceneId);
        }

}
```

Εικόνα 5.70 – Υλοποίηση της συνάρτησης OnDocumentMouseDown()

Η συνάρτηση onDocumentMouseDown() (εικόνα 5.70) λειτουργεί παρόμοια με την updateπου είδαμε παραπάνω. Ελέγχει αν στο σημείο το οποίο έγινε κλικ η προβολή προσπίπτει σε αντικείμενο που ανήκει στην targetList. Αν προσπίπτει, τότε ανάλογα με το επίπεδο που βρισκόμαστε γίνεται η αλλαγή προς το επόμενο επίπεδο. Στη συνέχεια διαγράφεται το περιεχόμενο της σκηνής μέσω της clearScene(), έπειτα

γίνεται η κλήση των συναρτήσεων clearTargetList(), clearSceneModels(), clearSceneGround() και clearScenePins() οι οποίες αποδεσμεύουν τις μεταβλητές οι οποίες χρησιμοποιούνται για την διατήρηση των δεδομένων της σκηνής, ούτως ώστε να μην γίνεται σπάταλη στη χρήση της μνήμης που χρησιμοποιεί ο φυλλομετρητής για την διατήρηση των μεταβλητών. Τέλος καλείται η συνάρτηση getAndPreviewSceneFromDb(intersects[0].object.sceneID) η οποία είναι η συνάρτηση που μελετήσαμε προηγουμένως για την ανάκτηση της σκηνής από τον server. Σαν όρισμα αυτή τη φορά παίρνει το idτου μοντέλου ground, το οποίο αποτελεί την βάση της κάθε σκηνής, το οποίο επιλέχθηκε από το κλικ του χρήστη μέσω της συνάρτησης onDocumentMouseDown().Αν η επιλογή μας οδηγεί στο επίπεδο Monument τότε εμφανίζουμε ένα βέλος ανοίγματος το οποίο μπορεί να επιλεχτεί ούτως ώστε να ανοίξει το hide Menuτο οποίο έχουμε υλοποιήσει μέσω κώδικα HTML. Κάθε φορά που γίνεται η επιλογή μίας από τις επιλογές στο menu τότε ανοίγει μία διάφανη καρτέλα μπροστά από την τρισδιάστατη σκηνή μας και εμφανίζεται το αντίστοιχο περιεχόμενο το οποίο έχει ανακτηθεί από την βάση δεδομένων.

6 Γραφική Διεπαφή

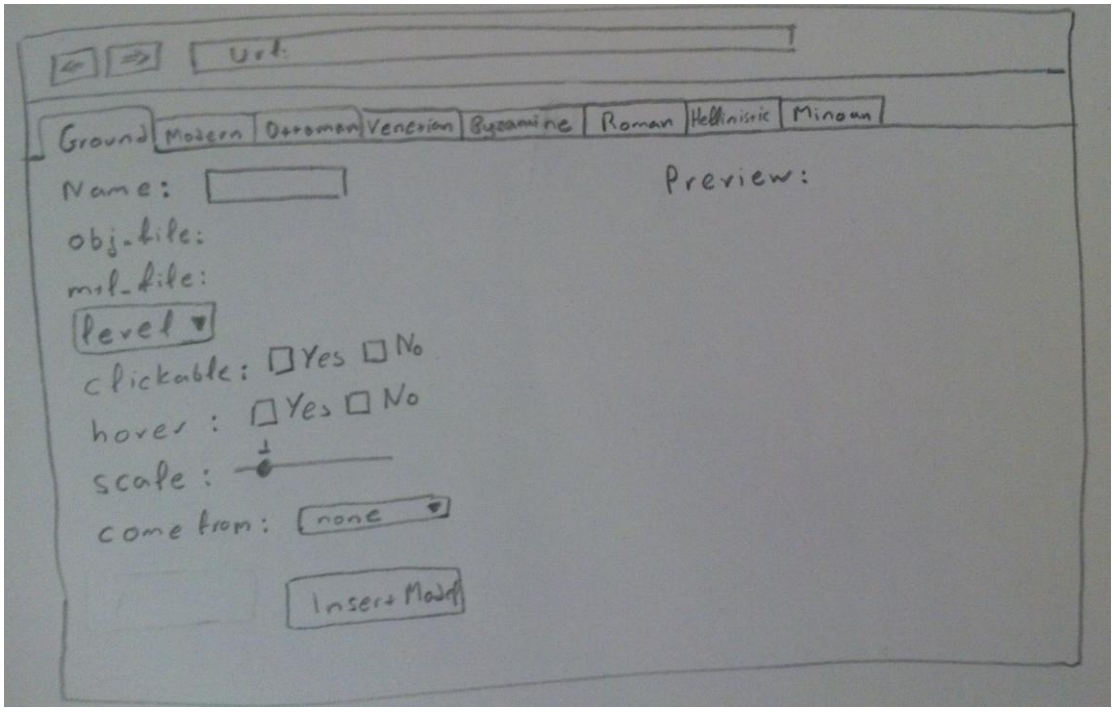
6.1 Εισαγωγή

Σε αυτό το κεφάλαιο παρουσιάζουμε την μεθοδολογία που ακολουθήθηκε για τον σχεδιασμό της γραφικής διεπαφής του χρήστη στην εφαρμογή-διαχείρισης και την εφαρμογή επισκέπτη καθώς επίσης και το τελικό αποτέλεσμα της εφαρμογής. Στην ενότητα 6.2 παρουσιάζεται η φάση σχεδιασμού μέσω paperprototypingενώ στην ενότητα 6.3 παρουσιάζεται η γραφική διεπαφή όπως αυτή παρουσιάζεται στους πραγματικούς χρήστες.

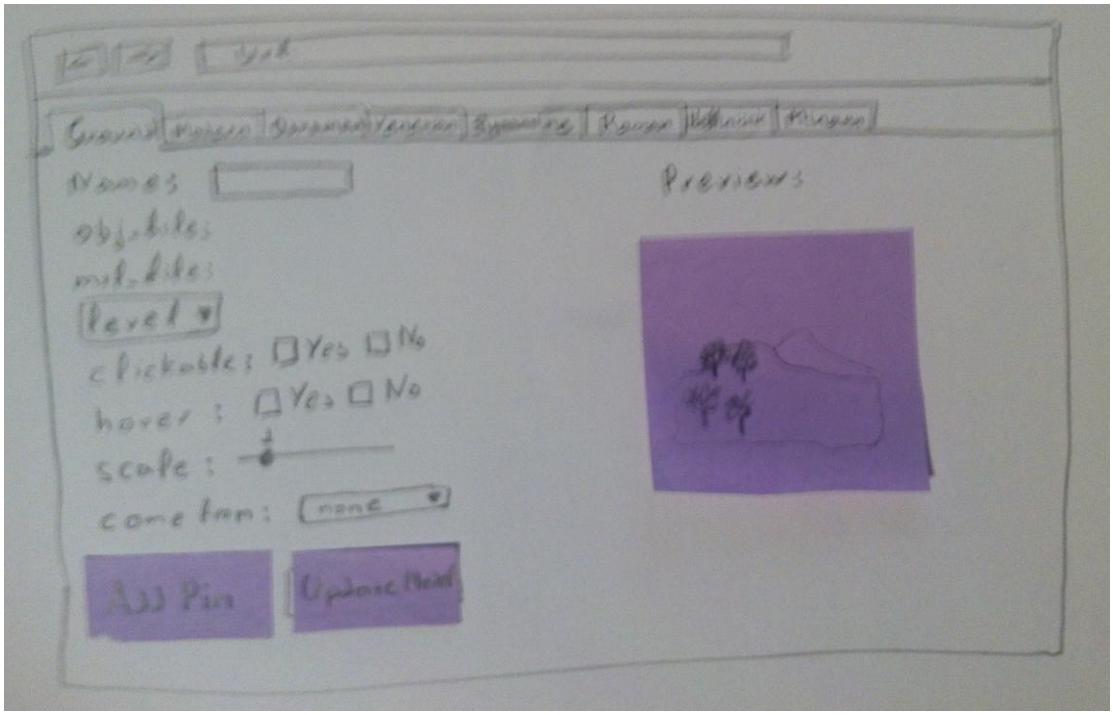
6.2 Σχεδιασμός γραφικής διεπαφής

Στα αρχικά στάδια σχεδιασμού της εφαρμογής, σχεδιάσαμε κάποια πρωτότυπα των γραφικών διεπαφών χρήστη και δημιουργήσαμε storyboardsγια να απεικονίσουμε και να οργανώσουμε τις ιδέες μας. Ένα storyboardείναι μία αναπαράσταση μίας συγκεκριμένης ακολουθίας αλληλεπίδρασης του χρήστη με την εφαρμογή μέσω των πρωτότυπων. Tostoryboardκαταστεί δυνατή την καλύτερη απεικόνιση της αλληλεπίδρασης των οθονών, ενώ παρουσιάζει το μεγαλύτερο μέρος της λειτουργικότητας του συστήματος. Επιπλέον είναι κατάλληλο για τον εντοπισμό προβλημάτων σχεδιασμού τα οποία μπορούν να λυθούν πολύ γρήγορα μέσω αυτής της μεθόδου.

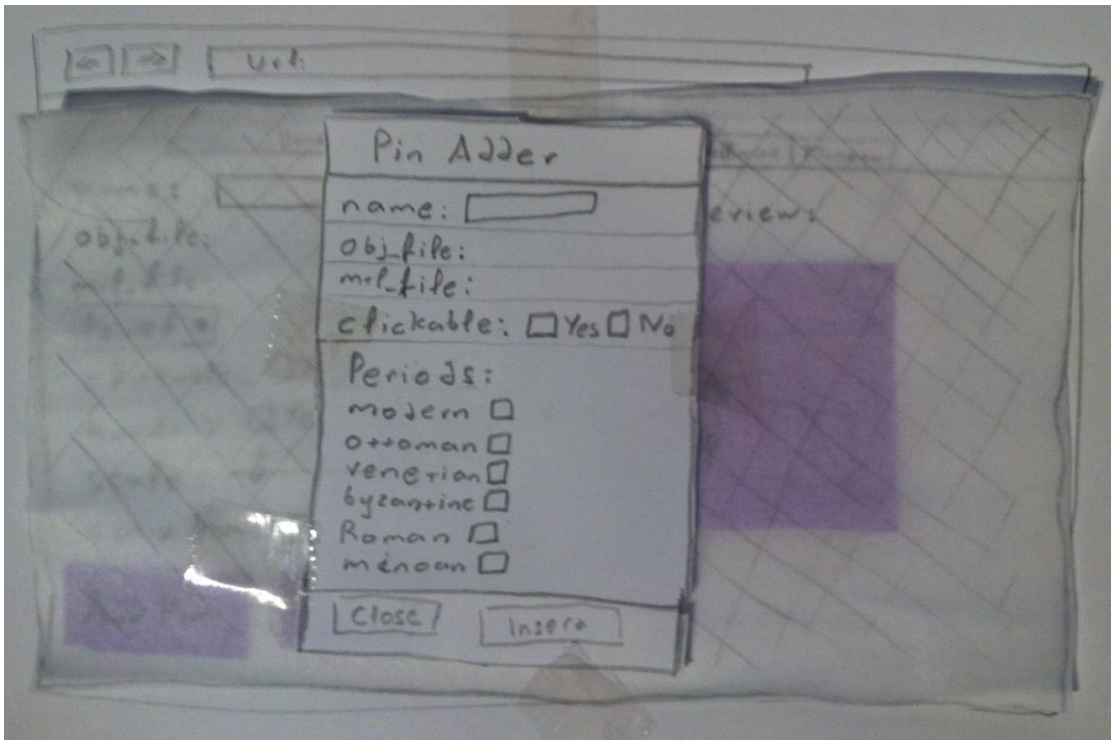
6.2.1 Εφαρμογή διαχείρισης



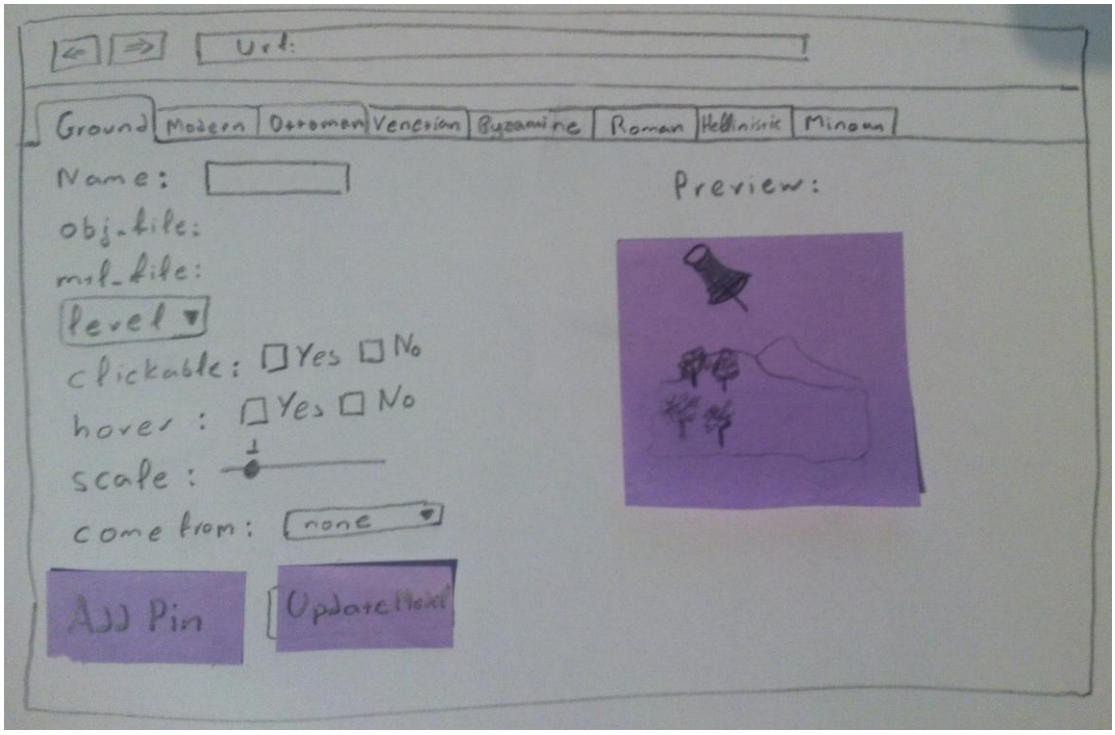
Εικόνα 6.1 – PaperPrototype: Οθόνη προβολής φόρμας για εισαγωγή μοντέλου Ground



Εικόνα 6.2 – Οθόνη επιτυχούς εισαγωγής του μοντέλου Groundκαι προεπισκόπησης του.

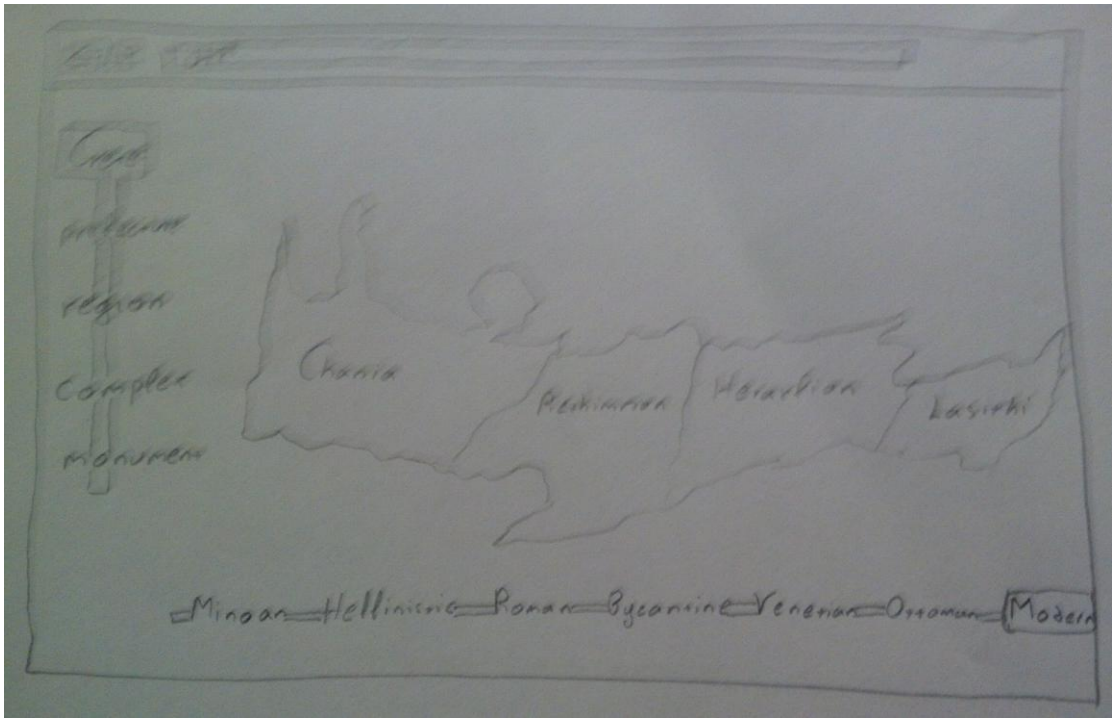


Εικόνα 6.3 – Οθόνη φόρμας εισαγωγής μοντέλου τύπου pin.

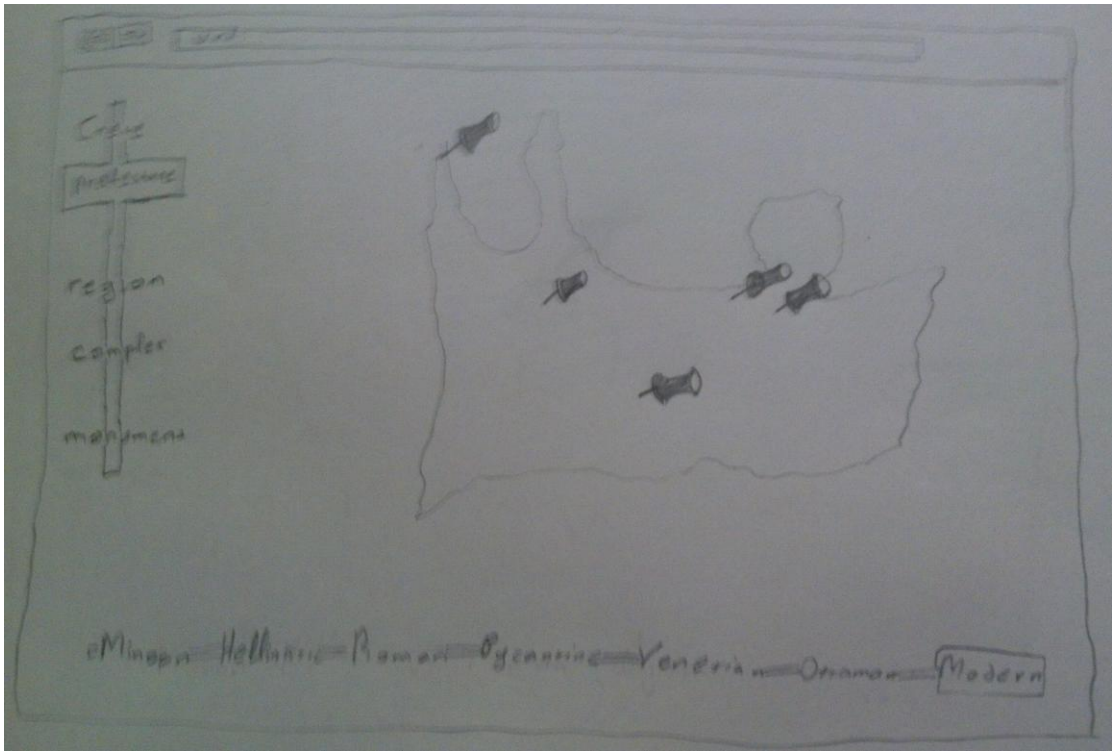


Εικόνα 6.4 – Οθόνη προεπισκόπησης σκηνής εδάφους μαζί με το pinπου προστέθηκε.

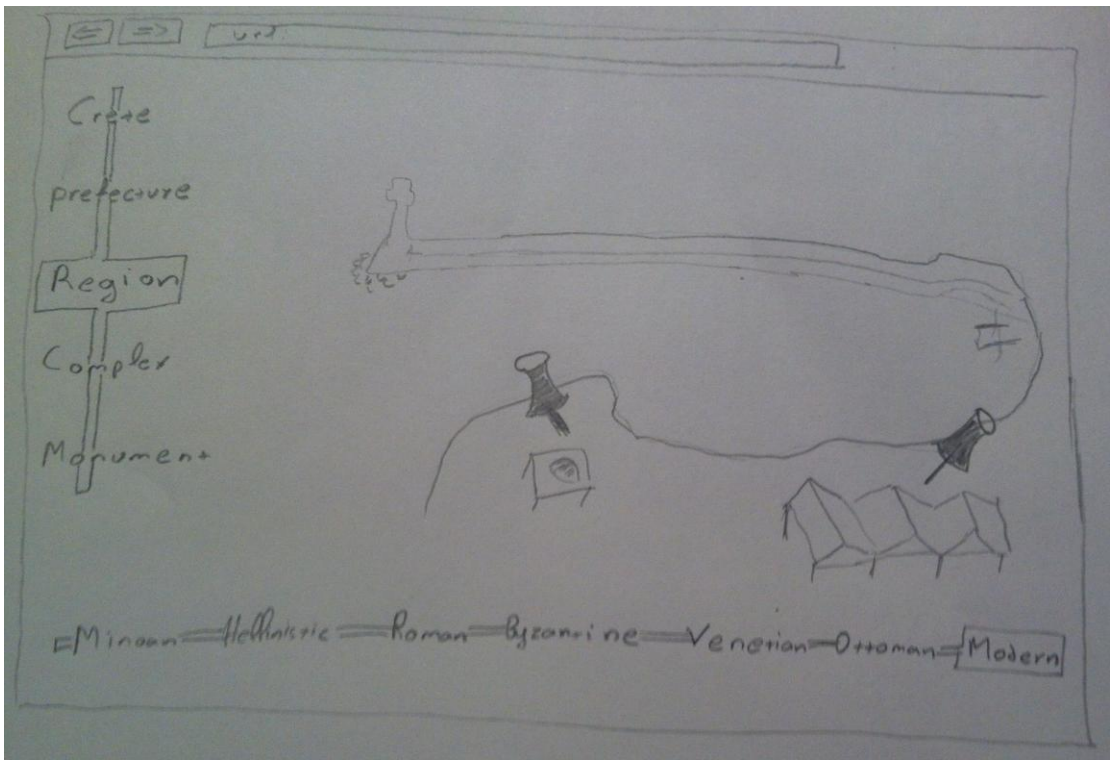
6.2.2 Εφαρμογή επισκέπτη



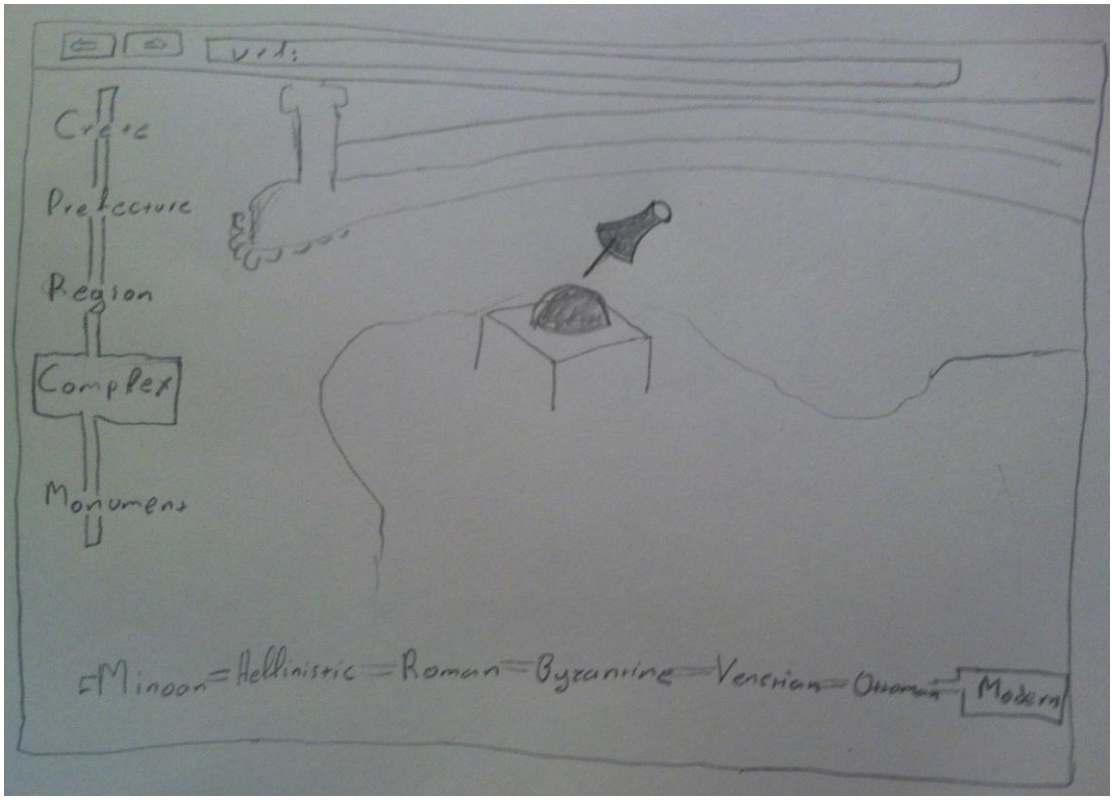
Εικόνα 6.5 – Οθόνη εισόδου στην εφαρμογή επισκέπτη, προβολή του επιπέδου Κρήτη



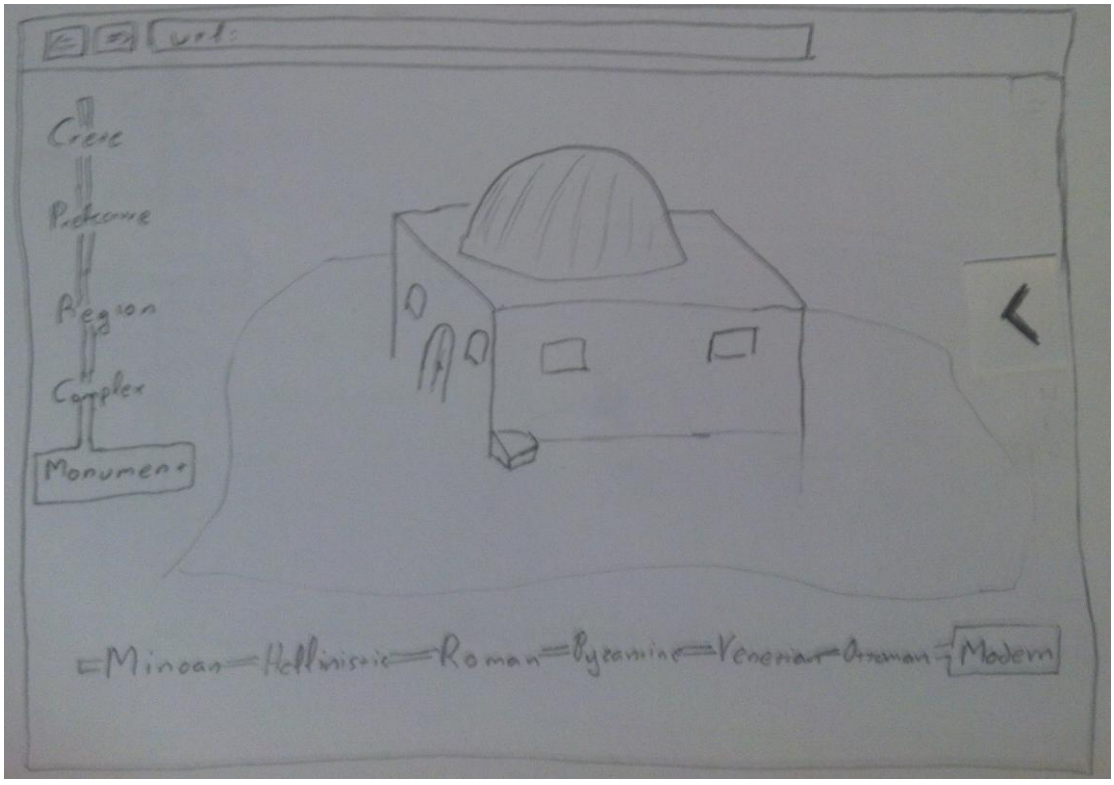
Εικόνα 6.6 – Οθόνη προβολής επιπέδου prefecture(Χανιά)



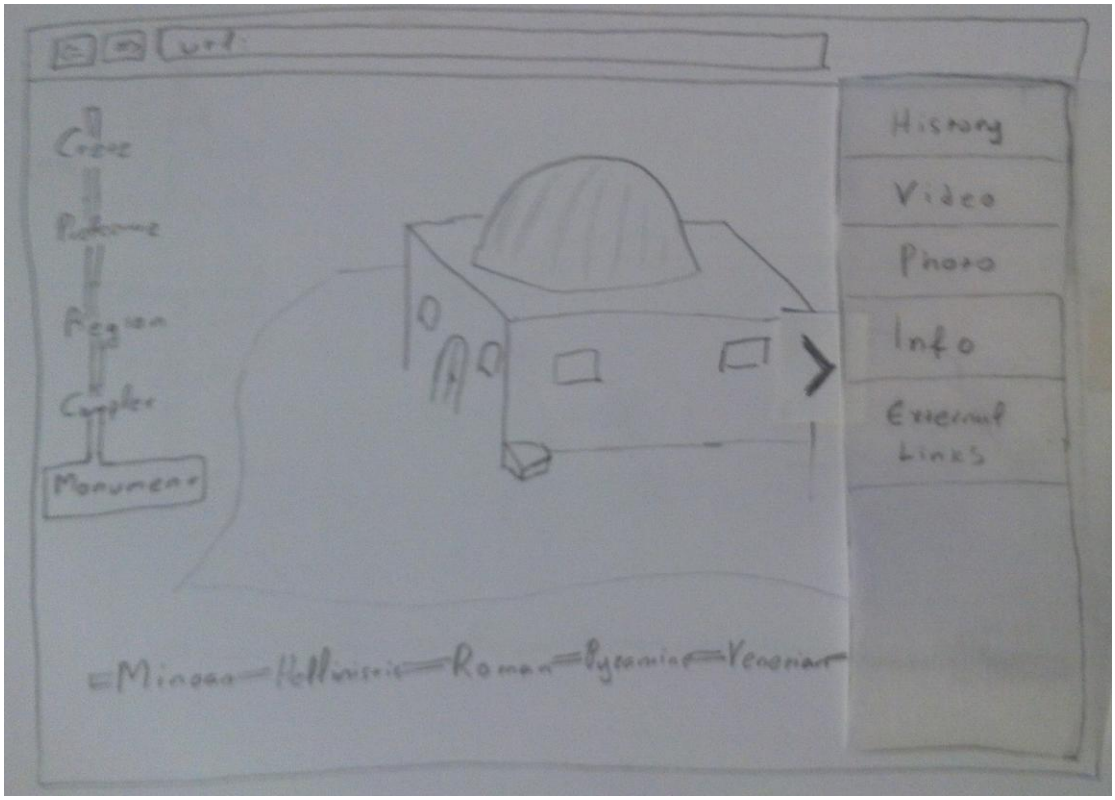
Εικόνα 6.7 – Οθόνη προβολής επιπέδου Region (Region of Giali tzami)



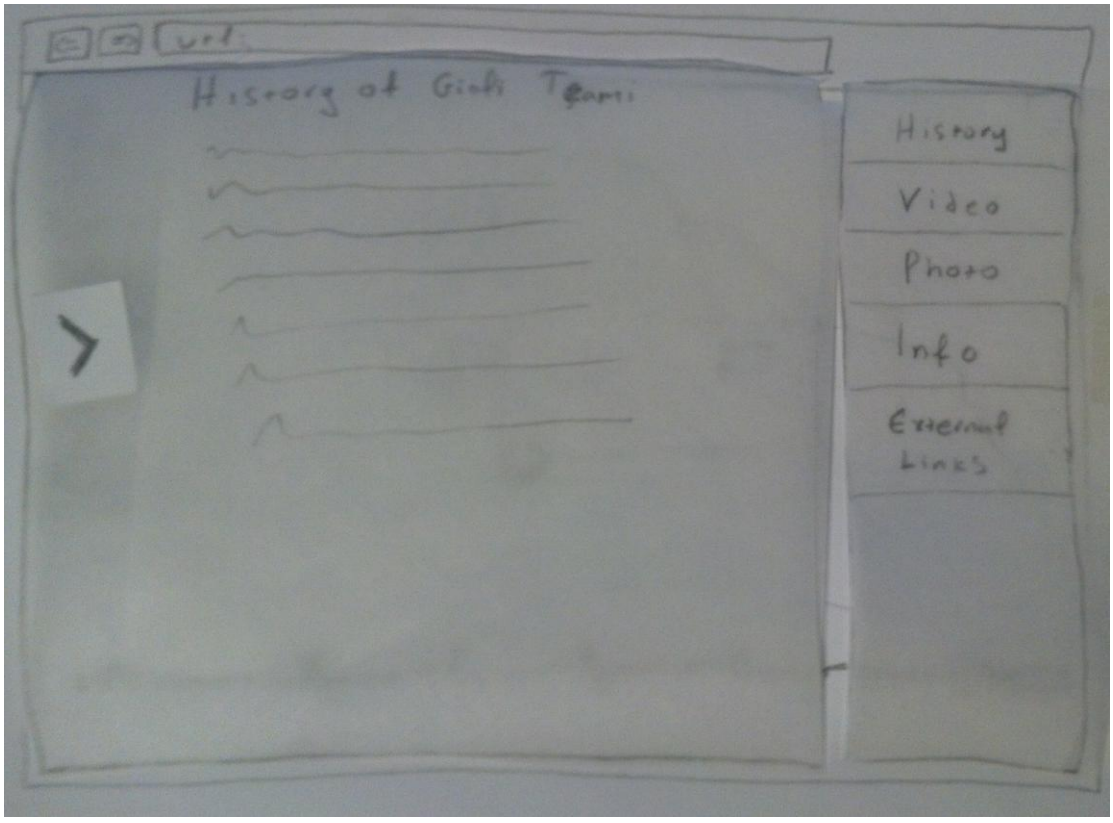
Εικόνα 6.8 – Οθόνη προβολής επιπέδου Complex (Complexofgialitzami)



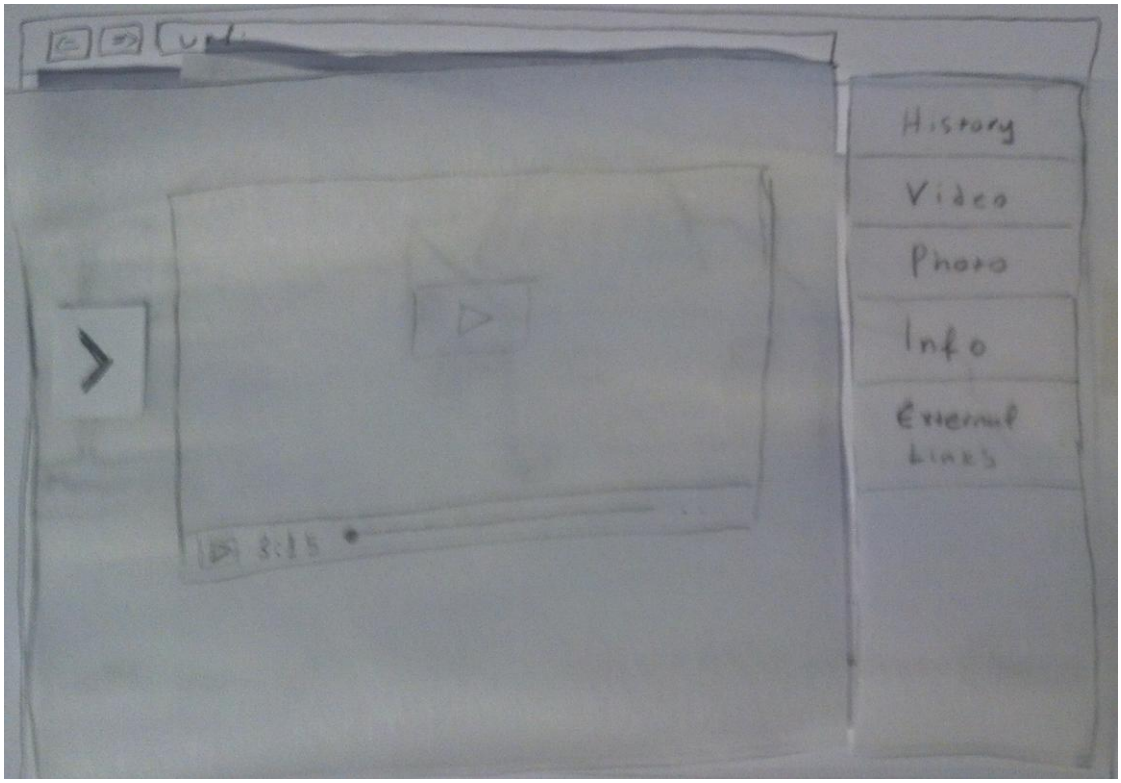
Εικόνα 6.9 – Οθόνη προβολής επιπέδου Monument (Γιαλί τζαμί)



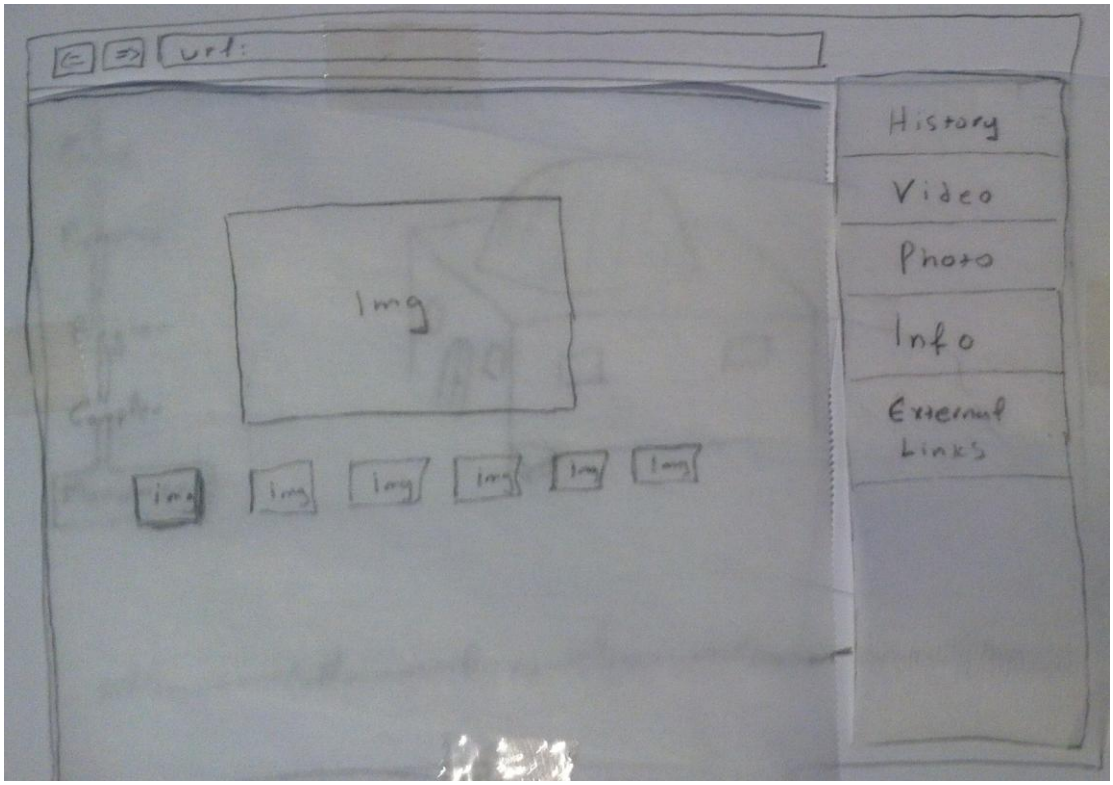
Εικόνα 6.10 – Προβολή Οθόνης επιπέδου Monumentμε ανοιχτό το κρυφό μενού



Εικόνα 6.11 – Οθόνη ανοίγματος καρτέλας των ιστορικών πληροφοριών



Εικόνα 6.12 – Οθόνη καρτέλας αναπαραγωγής βίντεο σχετικό με το μνημείο.



Εικόνα 6.13 – Οθόνη ανοίγματος καρτέλας προβολής σχετικών φωτογραφιών.

6.3 Αξιολόγηση εφαρμογής

Κατά την διάρκεια του σχεδιασμού της εφαρμογής μέσω της χρήσης των paper prototypes, έγινε αξιολόγηση η οποία μας οδήγησε στην τελική μορφή των paper prototypes μέσω της οποίας προέκυψε η γραφική διεπαφή που παρουσιάζεται στην επόμενη ενότητα. Να τονιστεί σε αυτό το σημείο ότι υπήρχε συνεχώς ανατροφοδότηση μέσω σχολίων από συμφοιτητές αλλά και όλη την ομάδα που ασχολήθηκε καθώς παρουσιάζαμε demo σε κάθε συνάντηση και μέσω των σχολίων και παρατηρήσεων γινόντουσαν συνεχώς απαραίτητες αλλαγές.

Για την αξιολόγηση και την εύρεση λαθών χρησιμοποιήθηκαν φοιτητές του τμήματος, στους οποίους ζητήθηκε να πάρουν τον ρόλο απλών χρηστών του ηλεκτρονικού υπολογιστή, μη θεωρώντας δεδομένες τις γνώσεις που έχουν αποκομίσει μέσω των σπουδών τους. Λόγω της πίεσης των παραδοτέων για την εκπλήρωση των χρονικών δεσμεύσεων απέναντι στην εταιρία που χρηματοδότησε την εφαρμογή, δεν ακολουθήθηκε συγκεκριμένο μοτίβο αξιολόγησης καθώς η εφαρμογή παρουσιάζει τρισδιάστατο περιεχόμενο και αυτό έκανε πιο δύσκολη την κατανόηση της στο χαρτί καθώς δεν μπορούσαμε να απεικονίσουμε πληροφορίες που εμφανίζονται περνώντας τον κέρσορα πάνω από τρισδιάστατα μοντέλα.

Ζητήθηκε από τους χρήστες να εκτελέσουν συγκεκριμένες διαδρομές μέσω των οποίων περνάνε από όλα τα βασικά στάδια της εφαρμογής. Πιο συγκεκριμένα το σενάριο που έπρεπε να ακολουθηθεί ήταν η επιλογή του νομού χανιών με σκοπό την επισκόπηση του μνημείου “Γιαλί Τζαμί” στο παλιό λιμάνι, και την προβολή όλων των επιμέρους πληροφοριών. Με τον τρόπο αυτό ορίσαμε ουσιαστικά ένα σενάριο, το οποίο όμως διαπερνά ολόκληρη την εφαρμογή. Ο ρόλος μας ήταν επεξηγηματικός κατά την διαδικασία καθώς έπρεπε να εξηγήσουμε ό,τι ότι βλέπουν είναι τρισδιάστατο και υπάρχει διαδραστικότητα με το περιεχόμενο. Ήμασταν ανοιχτοί σε κάθε αντίδραση, και δεκτικοί σε αρνητικά σχόλια καθώς αυτό θα βοηθούσε στον καλύτερο σχεδιασμό της εφαρμογής, και ζητήσαμε από τους χρήστες να μας προτείνουν τις ιδέες τους.

Τα συμπεράσματα από την αξιολόγηση μας έδειξαν λάθη τα οποία είχαν γίνει σχεδιαστικά και διορθώθηκαν άμεσα στα paper prototypes και εν συνεχεία σχεδιάστηκε με αυτόν τον τρόπο και η γραφική διεπαφή. Υπήρχε σαν γενική αντίδραση απορία με το τι εννοούσαμε τρισδιάστατο αντικείμενο, και πως θα το χειρίζεται ο χρήστης. Βασικές αλλαγές που έγιναν ήταν στον σχεδιασμό της κάθετης και οριζόντιας μπάρας, έτσι ώστε να είναι στην μορφή κουμπιών, ενώ αρχικά ήταν τύπου “χάρακα”. Επιπλέον ζητήθηκε από τους χρήστες να γίνονται όλες οι μεταβάσεις με την χρήση πινέζας, το οποίο σε ορισμένες μεταβάσεις δεν είχαμε με αποτέλεσμα να μην ξέρουν ποιο αντικείμενο είναι το ζητούμενο. Ένας εκ των χρηστών μας ζήτησε να εισάγουμε κείμενο με βοηθητικές εντολές, καθώς δεν είναι σύνηθες το περιεχόμενο τριών διαστάσεων. Τέλος στον αρχικό σχεδιασμό δεν ήταν διακριτός ο τρόπος με τον οποίο ανοίγει το κρυφό μενού, με αποτέλεσμα να χρησιμοποιήσουμε ένα πιο διακριτό σύμβολο με μεγαλύτερες διαστάσεις στην τελική εφαρμογή. Οι γενικές αντιδράσεις μετά το πέρας του σεναρίου στο χαρτί και της συζήτησης ήταν ενθαρρυντικές για την συνέχεια καθώς η πρωτοτυπία της εφαρμογής τράβηξε το ενδιαφέρον των χρηστών και μας κάναν ερωτήσεις εκτός σεναρίου, το οποίο μας έδωσε και ιδέες για μελλοντικές επεκτάσεις που

θα αναφέρουμε και σε επόμενο κεφάλαιο. Οι αλλαγές οι οποίες πραγματοποιήσαμε στις δύο μπάρες έδωσαν ένα πιο καλό αισθητικά αλλά και χρηστικά αποτέλεσμα σε σχέση με τον αρχικό σχεδιασμό. Η εισαγωγή βασικών οδηγιών της χρήσης ήταν κάτι το οποίο δεν θα έπρεπε να λείπει, και μας φάνηκε πολύ χρήσιμο σαν αποτέλεσμα. Τέλος μέσω της αξιολόγησης κάναμε πιο εμφανές το κρυφό μενού, το οποίο θα ήταν κρίμα να μην εντοπίσει ο χρήστης χάνοντας πολύτιμες πληροφορίες που του παρέχει η εφαρμογή.

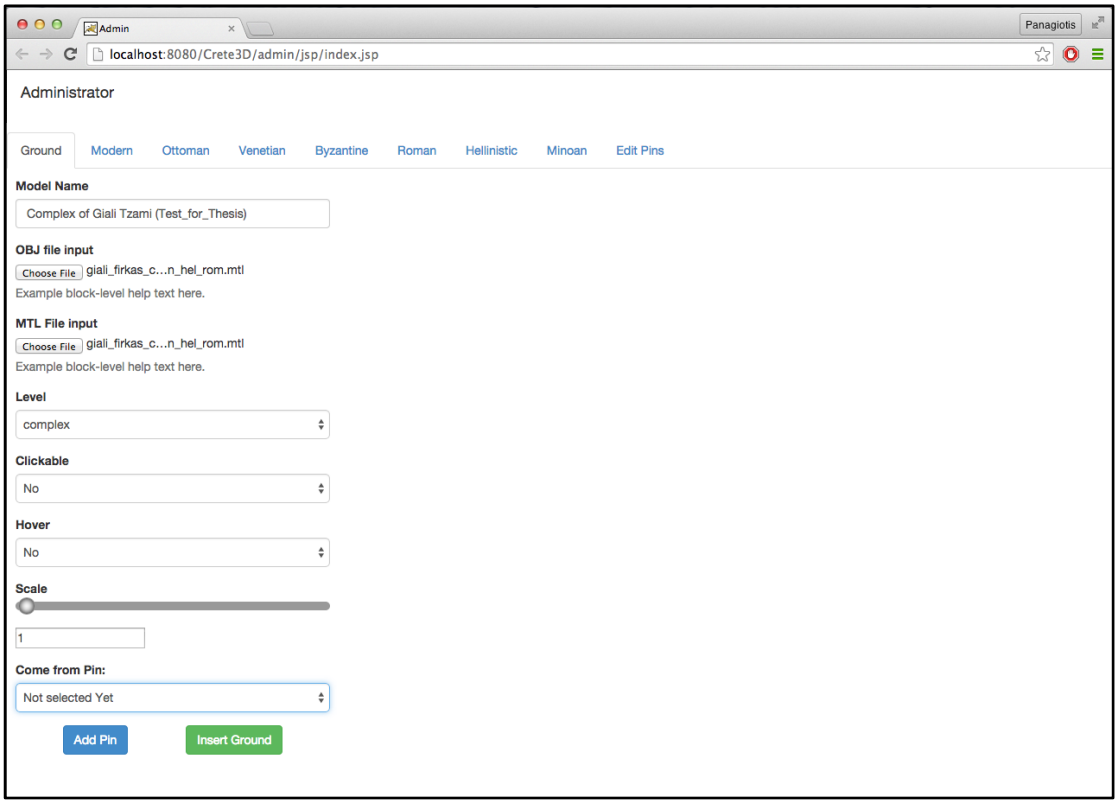
6.4 Γραφική διεπαφή εφαρμογής

Σε αυτή την ενότητα παρουσιάζονται μερικές από τις οθόνες της γραφικής διεπαφής χρήστη που εμφανίζονται στην τελική εφαρμογή, διαχείρισης και επισκέπτη, καθώς επίσεις περιγράφονται οι διεργασίες τις οποίες μπορεί να κάνει ο χρήστης από την εκάστοτε οθόνη.

6.4.1 Εφαρμογή Διαχείρισης

Εισαγωγή Εδάφους

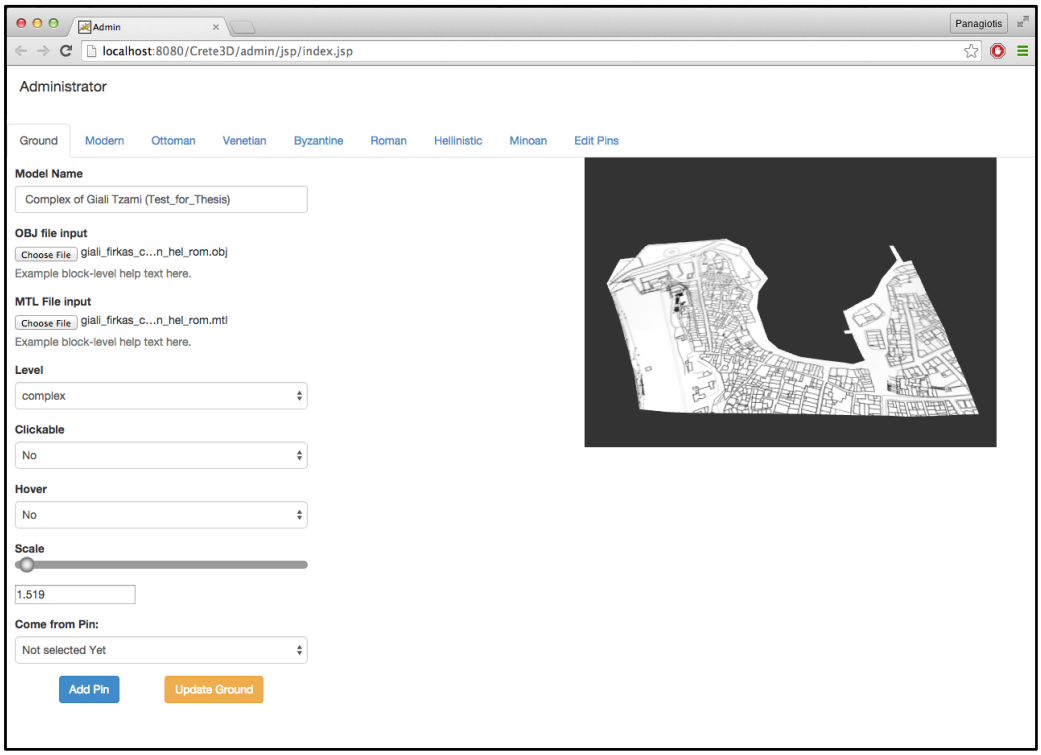
Όταν ο χρήστης διαχειριστής ανοίγει την εφαρμογή διαχείρισης εμφανίζεται η οθόνη της εικόνας 6.14. Υπάρχει η φόρμα εισαγωγής δεδομένων για την εισαγωγή εδάφους στην σκηνή που θέλει να υλοποιήσει. Συμπληρώνοντας τα απαραίτητα δεδομένα και ανεβάζοντας τα δύο αρχεία (myfile.obj, myfile.mtl) που συγκροτούν το τρισδιάστατο μοντέλο, και επιλέγοντας το κουμπί InsertGround, εισάγει το μοντέλο τύπου “Ground” στη σκηνή.



Εικόνα 6.14 – Οθόνη εισαγωγής μοντέλου τύπου εδάφους

Προεπισκόπηση εισαγωγής εδάφους

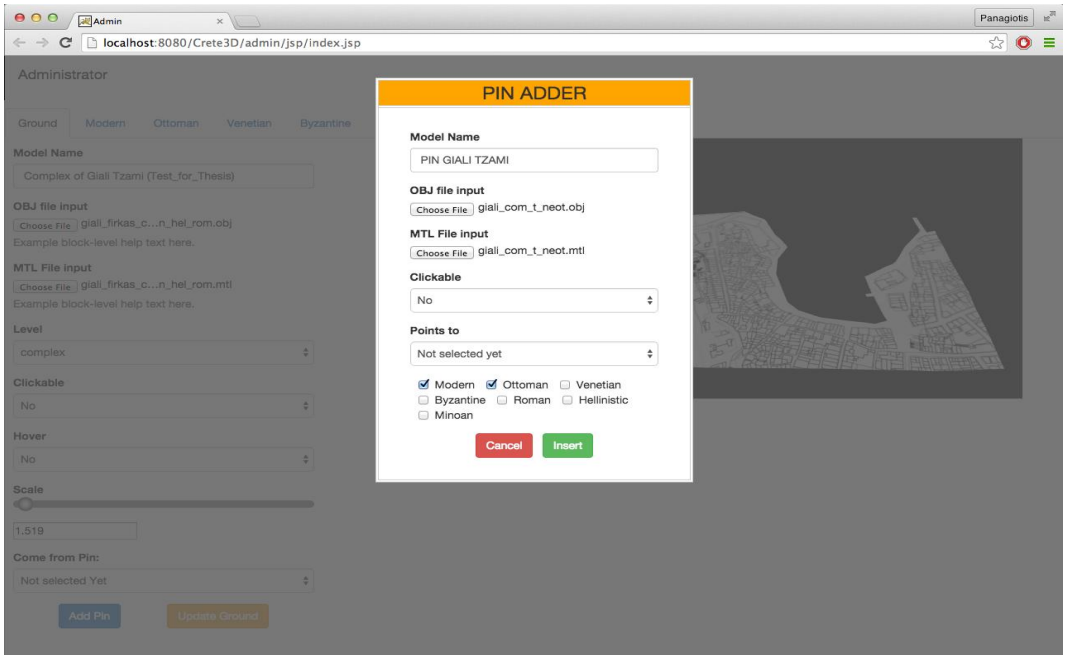
Στην οθόνη αυτή (εικόνα 6.15) ο χρήστης βλέπει τα στοιχεία που εισήγαγε στην βάση δεδομένων μέσω του μοντέλου εδάφους, καθώς επίσης και προεπισκόπηση του τρισδιάστατου μοντέλου που εισήγαγε. Μπορεί να αλλάξει τα δεδομένα του και να επιλέξει το κουμπί ”UpdateModel” μέσω του οποίου ανανεώνεται το μοντέλο που εισήγαγε. Επιπλέον μπορεί να προχωρήσει στην εισαγωγή Pin στην σκηνή του, καθώς επίσης και στις επόμενες καρτέλες εισαγωγής τρισδιάστατων μοντέλων ανάλογα με την εποχή.



Εικόνα 6.15 - Οθόνη προεπισκόπησης μοντέλου τύπου εδάφους

Οθόνη εισαγωγής PIN

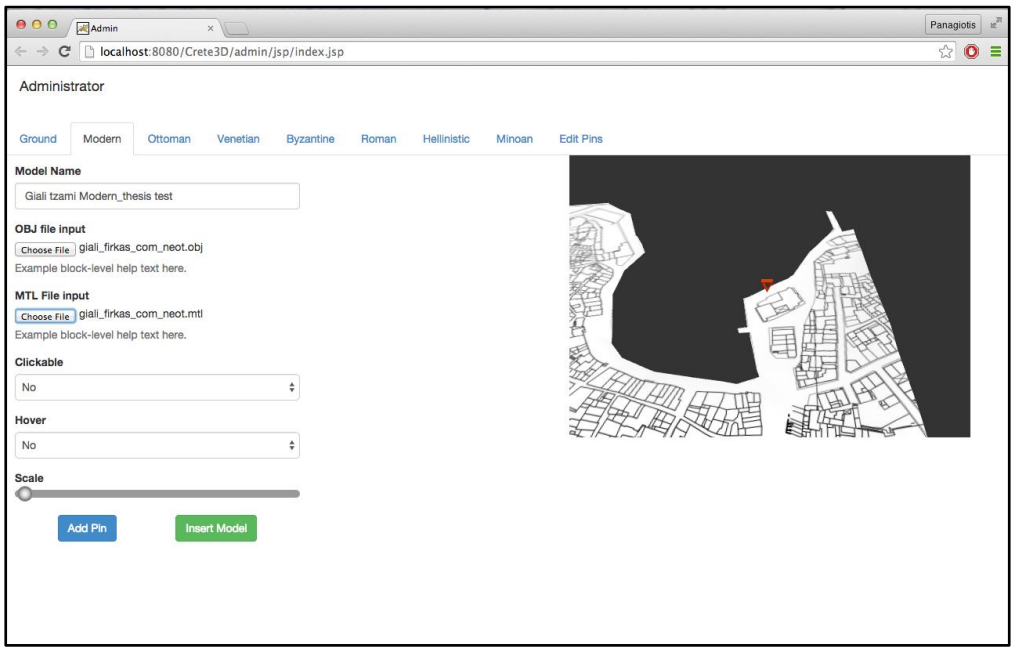
Σε αυτή την οθόνη (εικόνα 6.16) ο χρήστης έρχεται μέσω της επιλογής του κουμπιού ”AddPin”. Η οθόνη αυτή προβάλει την φόρμα εισαγωγής Pin. Ο χρήστης μπορεί να πατήσει cancelαν βρέθηκε σε αυτό το σημείο χωρίς να το θέλει, ή να εισάγει τα δεδομένα του pin, και να επιλέξει Insert, το οποίο θα εισάγει το pinστην σκηνή, στις επιλεγμένες από τον χρήστη εποχές.



Εικόνα 6.16 – Οθόνη εισαγωγής τρισδιάστατου αντικειμένου, τύπου PIN

Οθόνη εισαγωγής μοντέλου Νεότερης περιόδου

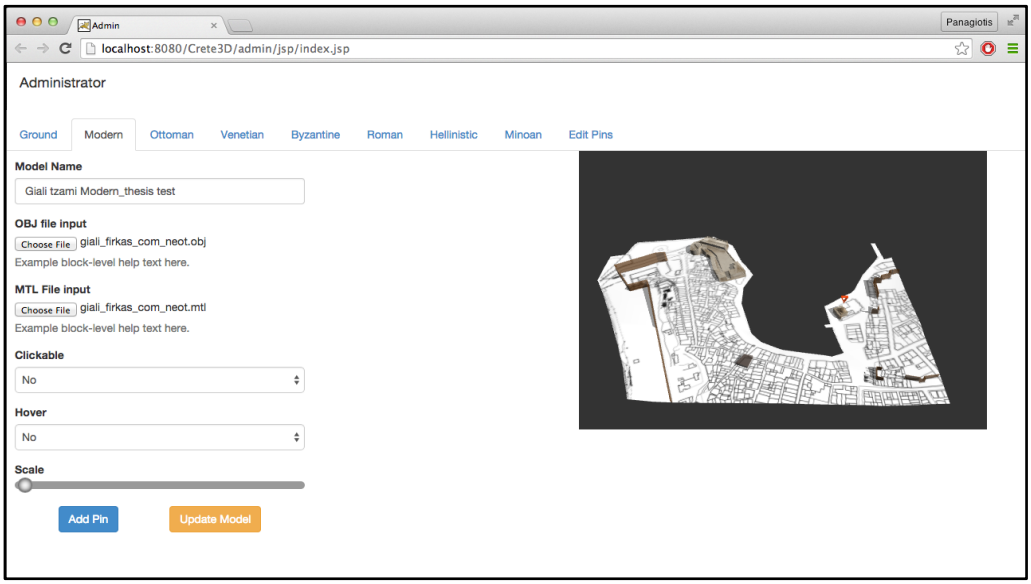
Η οθόνη αυτή (εικόνα 6.17) περιλαμβάνει την φόρμα εισαγωγής μοντέλου το οποίο ο χρήστης θέλει να εισάγει στην Ιστορική περίοδο Modern. Επιπλέον γίνεται προεπισκόπηση της σκηνής όπως είναι διαμορφωμένη μέχρι στιγμής για την ιστορική αυτή περίοδο. Δηλαδή εμφανίζεται το έδαφος που έχει εισαχθεί καθώς επίσης και όσα pinαντιστοιχούν σε αυτή τη περίοδο. Ο χρήστης συμπληρώνοντας την φόρμα εισαγωγής δεδομένων, και επιλέγοντας το κουμπί InsertModel, προσθέτει το μοντέλο που θέλει στην υπάρχουσα σκηνή, αλλά για την περίοδο Modern.



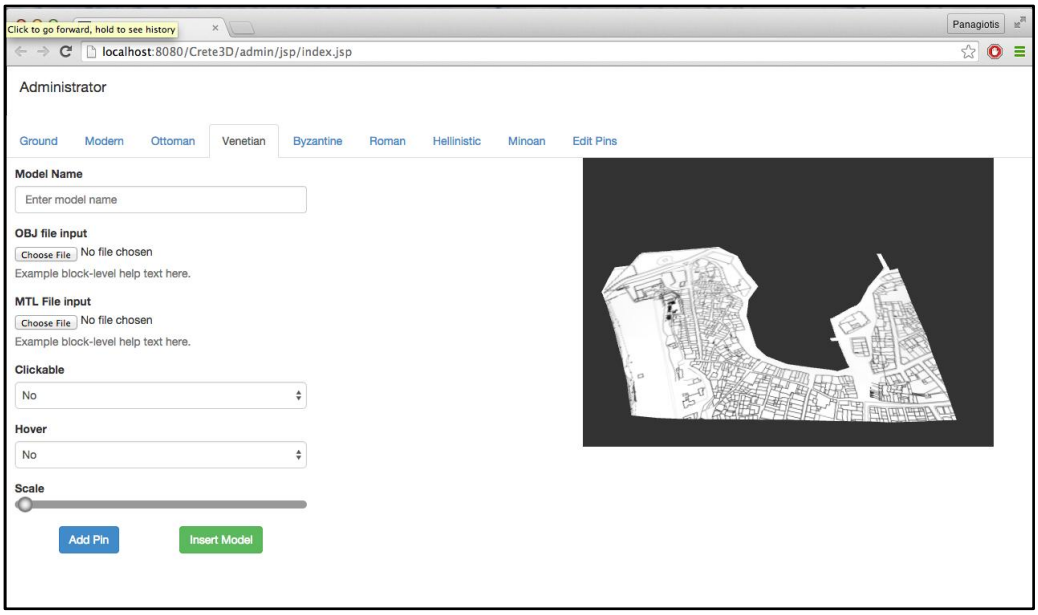
Εικόνα 6.17 – Οθόνη εισαγωγής τρισδιάστατου μοντέλου ιστορικής περιόδου modern.

Προεπισκόπηση εισαγωγής μοντέλου εποχής

Η οθόνη αυτή (εικόνα 6.18) περιέχει την φόρμα με την οποία ο χρήστης εισήγαγε τα δεδομένα του τρισδιάστατου μοντέλου και του επιτρέπει να ανανεώσει το μοντέλο του αν δεν τον ικανοποιεί το αποτέλεσμα της προεπισκόπησης της σκηνής. Επιπλέον στην προεπισκόπηση βλέπει την πλήρη σκηνή ανάλογα με την εποχή και έχει την δυνατότητα εναλλάσσοντας εποχές(εικόνα 6.19) να δεί την ώρα διαχείρισης την ιστορική εξέλιξη του μοντέλου του, ούτως ώστε να ελέγξει αν τον ικανοποιεί το αποτέλεσμα που θα προβάλλεται στον χρήστη-επισκέπτη.



Εικόνα 6.18- Οθόνη προεπισκόπησης σκηνής ιστορικής περιόδου modern

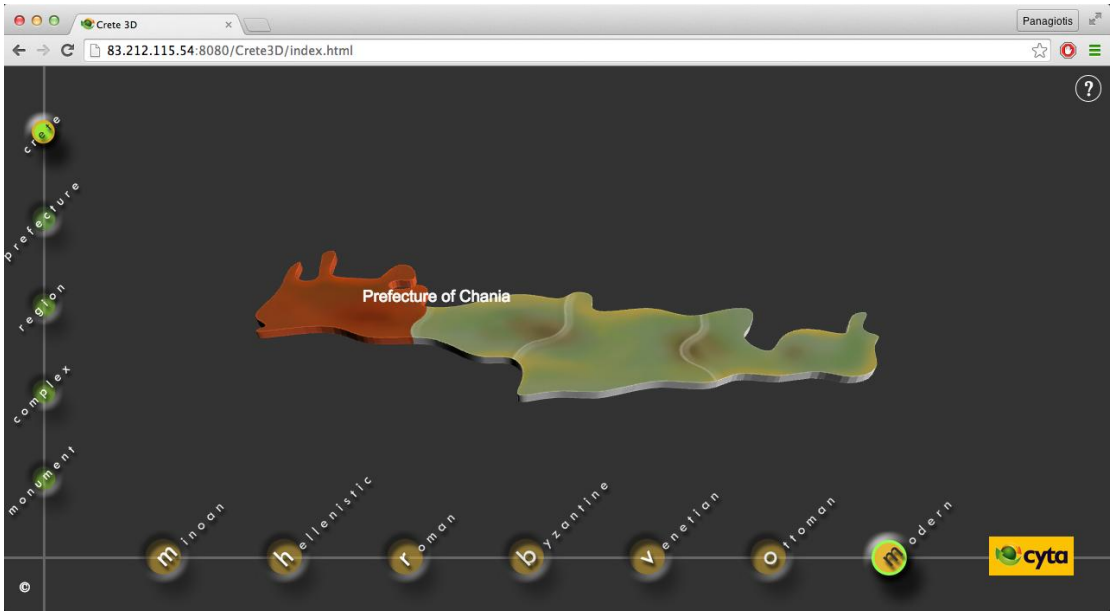


Εικόνα 6.19 - Οθόνη προεπισκόπησης σκηνής ιστορικής περιόδου Venetian

6.4.2 Εφαρμογή επισκέπτη

Οθόνη Επιπέδου ”Crete”

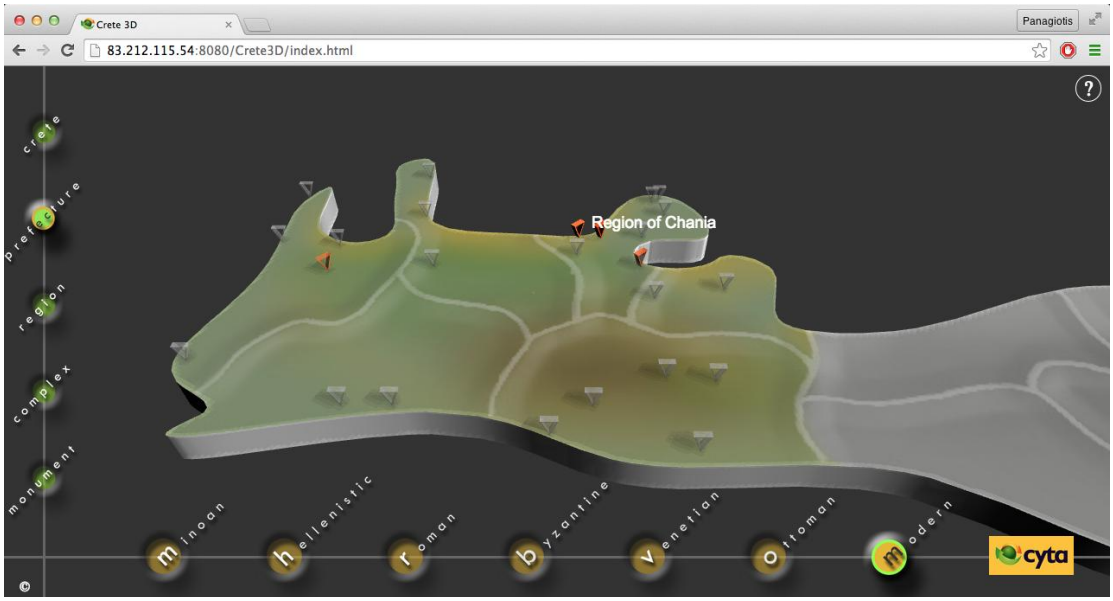
Με την είσοδο του χρήστη επισκέπτη στην εφαρμογή, εμφανίζεται η οθόνη της εφαρμογής που προβάλει τη σκηνή επιπέδου Crete (εικόνα 6.20). Στην οθόνη αυτή ο χρήστης μπορεί να αλληλοεπιδράσει με την τρισδιάστατη σκηνή μετακινώντας και περιστρέφοντας την καθώς επίσης να επιλέξει έναν από τους τέσσερις νομούς της Κρήτης ώστε να μεταβεί στο επίπεδο Prefecture. Μπορεί επίσης να αλλάξει ιστορική περίοδο, χωρίς όμως να πραγματοποιηθεί κάποια αλλαγή σε αυτό το επίπεδο, καθώς δεν αλλάζει η γεωγραφία της Κρήτης.



Εικόνα 6.20 – Οθόνη επιπέδου Crete

Οθόνη Επιπέδου ”Prefecture”

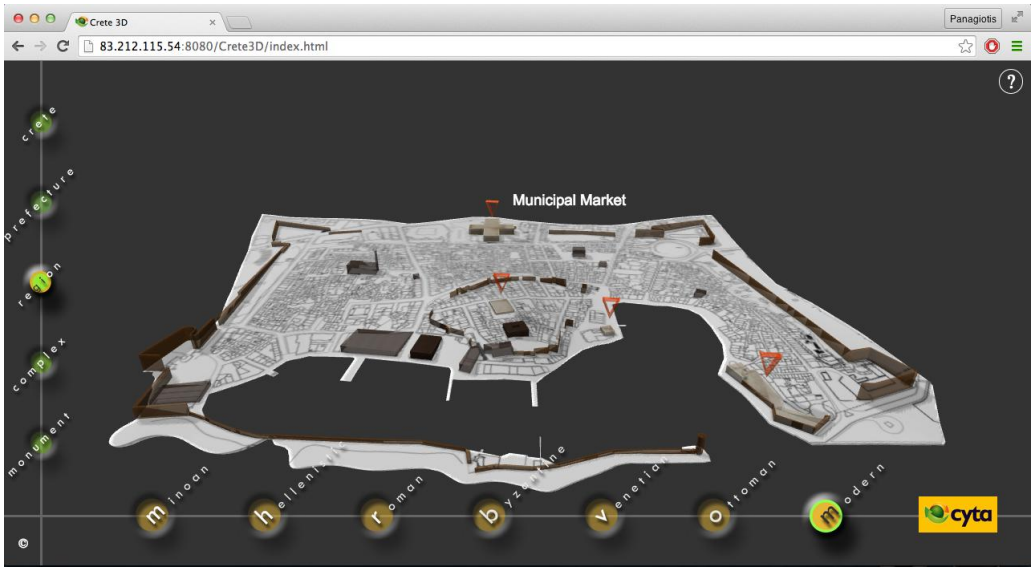
Ο χρήστης επιλέγοντας έναν εκ των νομών του επιπέδου Crete, μεταβαίνει στο επίπεδο Prefecture του νομού αυτού. Ο χρήστης σε αυτή την οθόνη (εικόνα 6.21) βλέπει τον επιλεγμένο νομό καθώς επίσης και Pins τα οποία υποδηλώνουν την ύπαρξη μνημείου στην σχετική θέση του κάθε Pin. Περνώντας τον κέρσορα πάνω από τα pinsεμφανίζεται η πιθανή μετάβαση που μπορεί να γίνει επιλέγοντας το. Τα πορτοκαλί συμπαγή Pins δηλώνουν ότι το μνημείο κατασκευάστηκε την εποχή στην οποία βρίσκεται ο χρήστης. Τα διάφανα πορτοκαλί δηλώνουν ότι το μνημείο κατασκευάστηκε σε προηγούμενη από την επιλεγμένη ιστορική περίοδο, ενώ τα γκρίζα ότι δεν έχει πραγματοποιηθεί περεταίρω υλοποίηση για το μνημείο αυτό. Σε αυτό το επίπεδο όπως και στα επόμενα, επιλέγοντας ιστορικές περιόδους μπορούμε να κατανοήσουμε πότε κατασκευάστηκε το κάθε μνημείο.



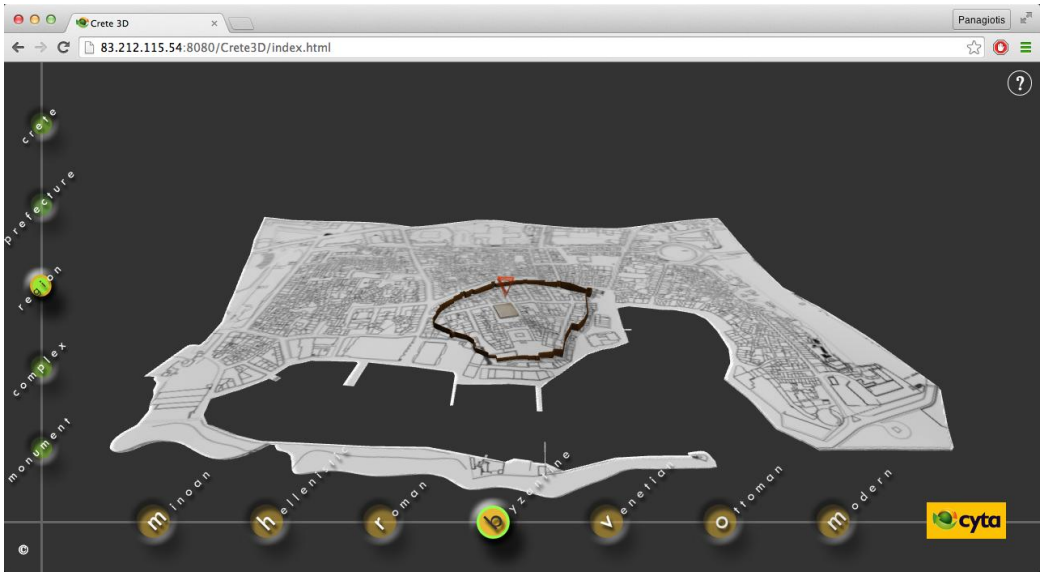
Εικόνα6.21 – ΟθόνηεπιπέδουPrefecture (Chania)

Οθόνη Επιπέδου ”Region”

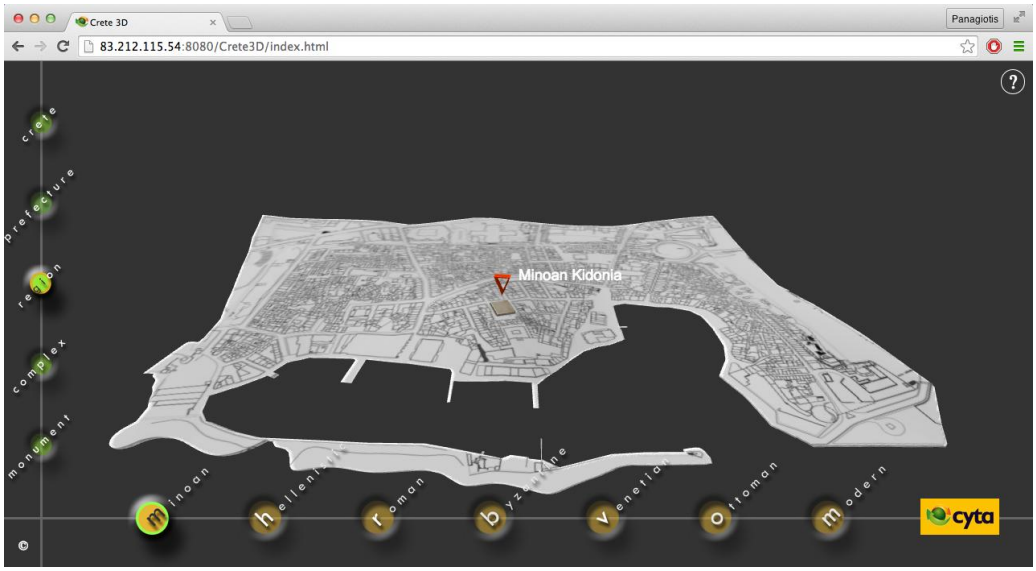
Ο χρήστης επιλέγοντας ένα από τα pinstου επιπέδου Prefectureμεταβαίνει στο επίπεδο Region. Σε αυτή την οθόνη ο χρήστης βλέπει την τρισδιάστατη σκηνή του επιπέδου Region. Δηλαδή βλέπει τον οικισμό ο οποίος μπορεί να περιλαμβάνει ένα ή περισσότερα μνημεία. Το τρισδιάστατο μοντέλο είναι σε αφαιρετικό σχεδιασμό και δεν περιέχει λεπτομέρεια. Κάθε μνημείο του οικισμού μπορεί να επιλεχτεί μέσω του Pin που το αντιπροσωπεύει. Ο χρήστης σε αυτό το επίπεδο, εναλλάσσοντας ιστορικές περιόδους μπορεί να δεί διαφορές στην εξέλιξη του οικισμού καθώς και αφαιρετικά στα μνημεία που τον αποτελούν. (εικόνα 6.22,6.23,6.24).



Εικόνα6.22 - ΟθόνηεπιπέδουRegion (Modern period)



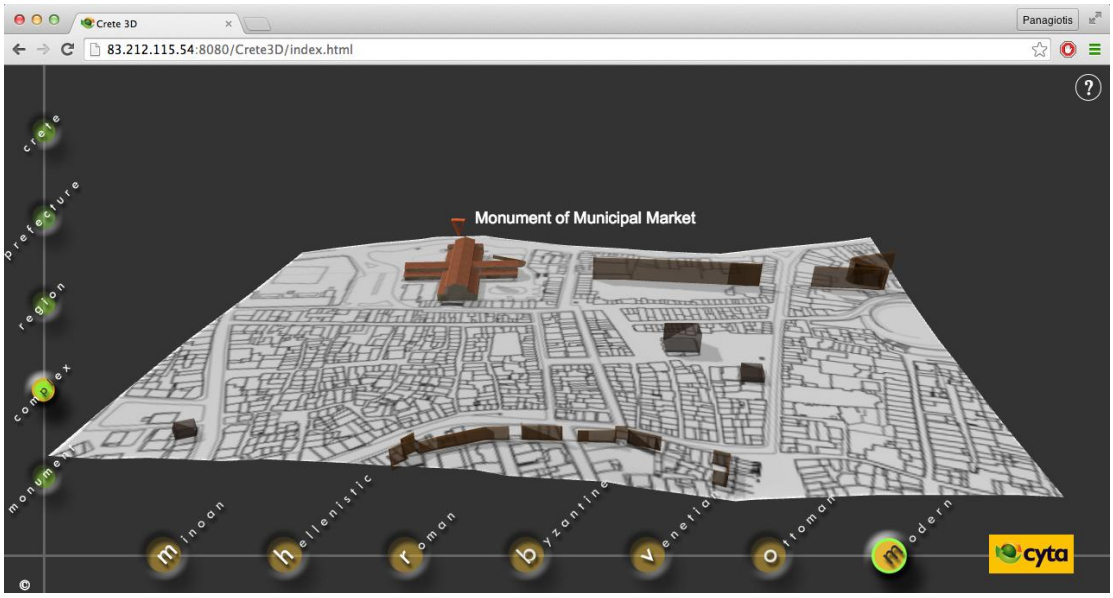
Εικόνα6.23- ΟθόνηεπιπέδουRegion (Byzantine period)



Εικόνα6.24ΟθόνηεπιπέδουRegion (Minoan period)

Οθόνη Επιπέδου ”Complex”

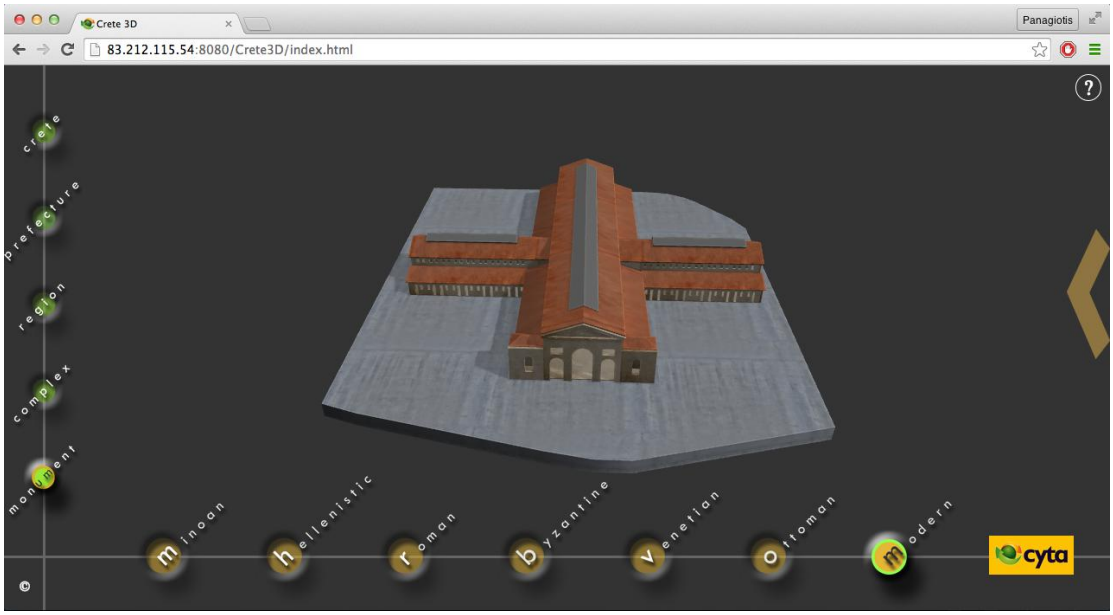
Ο χρήστης επιλέγοντας ένα Pintου επιπέδου Regionμεταβαίνει στο επίπεδο Complex, στο οποίο βλέπει το κτηριακό συγκρότημα που συγκροτεί το μνημείο (εικόνα 6.25). Σε αυτό το επίπεδο η λεπτομέρεια είναι περισσότερη από το επίπεδο Region. Επιλέγοντας ένα Pinσε αυτό το επίπεδο, ο χρήστης μεταβαίνει στο επόμενο επίπεδο το οποίο είναι το επίπεδο Monument.



Εικόνα 6.25 - Οθόνη επιπέδου Complex

Οθόνη Επιπέδου ”Monument”

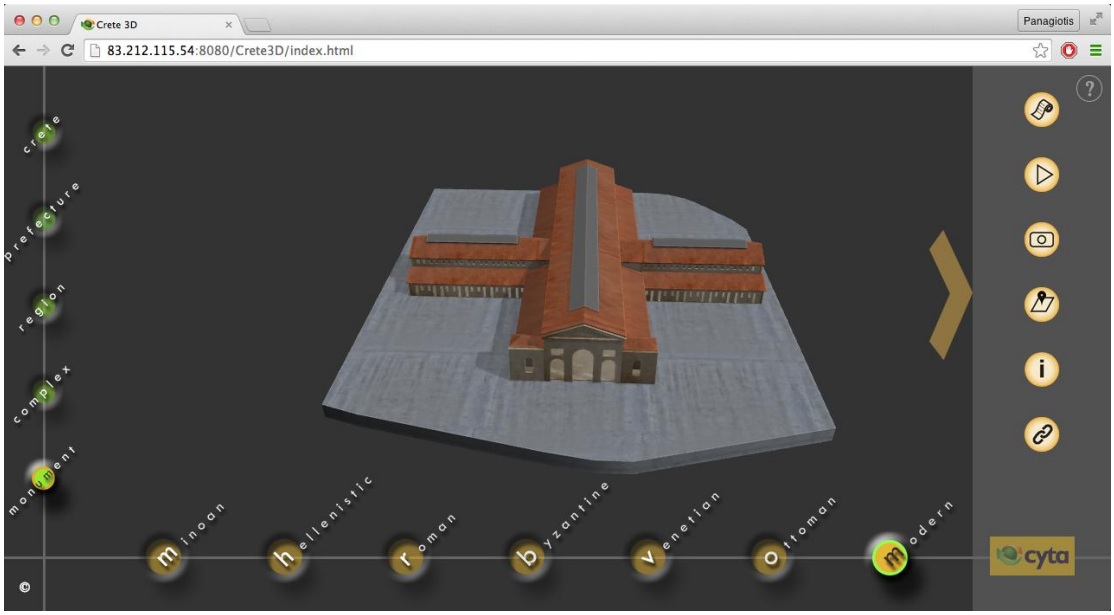
Ο χρήστησε αυτή την οθόνη (εικόνα 6.26) βλέπει το τελευταίο επίπεδο λεπτομέρειας του μνημείου. Μπορεί και σε αυτό το επίπεδο όπως και στα υπόλοιπα να το περιστρέψει και να το μετακινήσει. Επιπλέον σε αυτό το επίπεδο υπάρχει στα δεξιά ένα βέλος το οποίο επιλέγοντας το ανοίγει ένα αναδυπλώμενο μενού με περεταίρω πληροφορίες για το μνημείο.



Εικόνα 6.26 - Οθόνη επιπέδου Monument

Οθόνη Επιπέδου ”Monument” – Ανοιχτό μενού

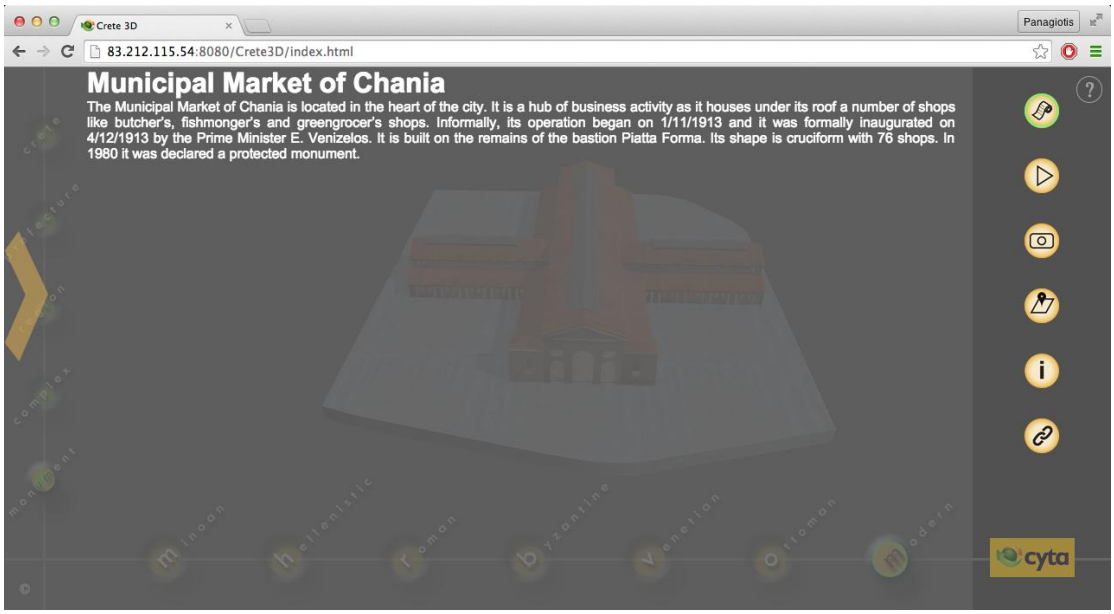
Ο χρήστησε αυτή την οθόνη (εικόνα 6.27) βλέπει το τελευταίο επίπεδο λεπτομέρειας έχοντας ανοίξει το μενού περισσοτέρων πληροφοριών.



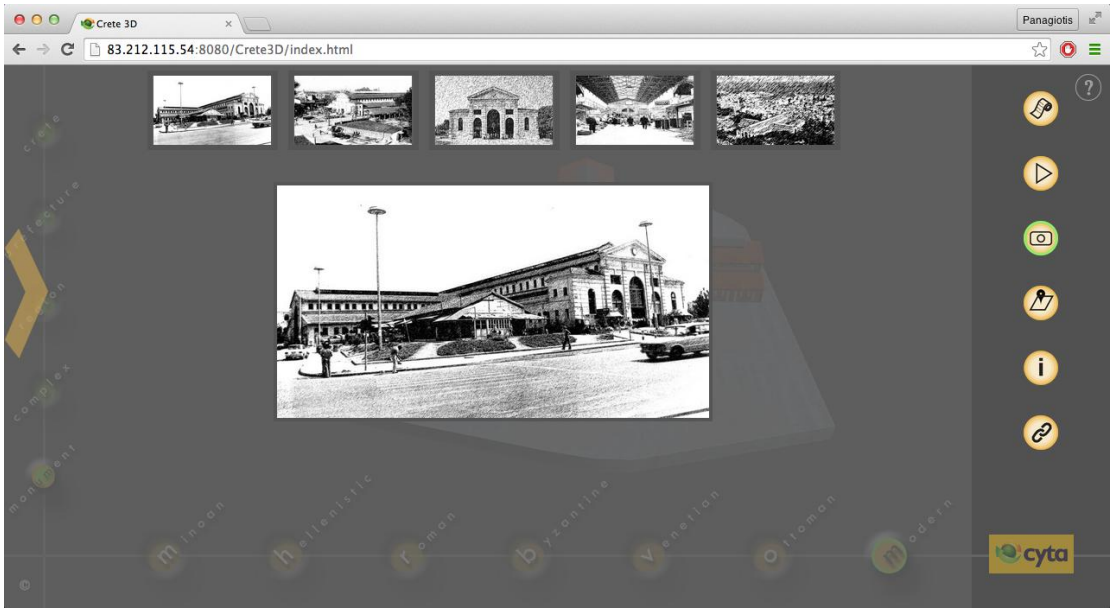
Εικόνα 6.27 Οθόνη Επιπέδου Monument- OpenMenu

Οθόνη Εμφάνισης Πληροφοριώνσχετικών με το μνημείο

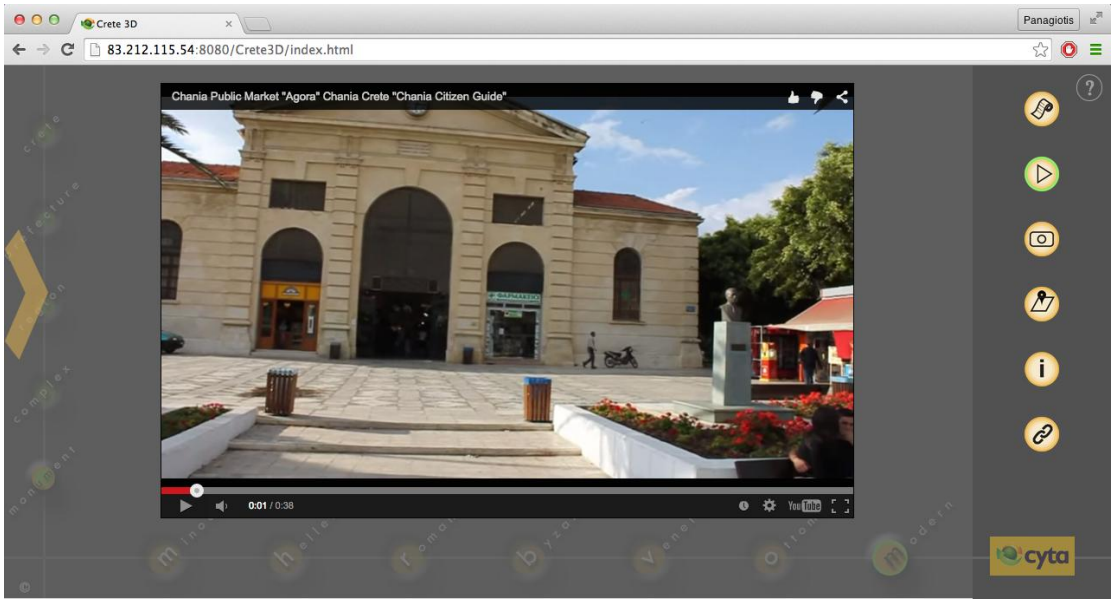
Ο χρήστης σε αυτή την οθόνη βλέπει τις πληροφορίες που έχει επιλέξει από το μενού. Μέσω του μενού μπορεί να επιλέξει να δεί ιστορικές πληροφορίες του μνημείου, σχετικό βίντεο, φωτογραφίες του μνημείου, χάρτη, επιπλέον πληροφορίες και εξωτερικούς συνδέσμους που το αφορούν. (εικόνες 6.28, 6.29, 6.30)



Εικόνα 6.28 – Οθόνη προβολής ιστορικών πληροφοριών μνημείου.



Εικόνα 6.29 – Οθόνη προβολής εικόνων σχετικών με το μνημείο



Εικόνα 6.30 – Οθόνη προβολής βίντεο σχετικό με το μνημείο

7 Ανακεφαλαίωση - Μελλοντικές Επεκτάσεις

7.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα κάνουμε ανακαιφαλαίωση της εφαρμογής που υλοποιήθηκε για την διπλωματική αυτή εργασία και θα αναφέρουμε μελλοντικές επεκτάσεις οι οποίες μπορούν να εξελίξουν την εφαρμογή.

7.2 Στόχοι και αποτελέσματα

Όπως έχουμε αναφέρει και σε προηγούμενα κεφάλαια, ο στόχος της διπλωματικής αυτής εργασίας ήταν η επιτυχής ανάπτυξη μίας διαδικτυακής εφαρμογής απεικόνισης τρισδιάστατων μνημείων της Κρήτης μέσω του φυλλομετρητή. Η εφαρμογή αυτή ήταν μέρος ενός έργου χρηματοδοτούμενου από την εταιρία CytaHellasπου έπρεπε να παραδοθεί εντός δέκα μηνών από την έναρξη της με ενδιάμεσα παρουσιάσιμα demo. Για την υλοποίηση της συνεργαστήκαμε με το τμήμα της αρχιτεκτονικής σχολής, από το οποίο προέκυψε μια πολυμελής ομάδα υλοποίησης. Ένας από τους στόχους λοιπόν ήταν και η επίτευξη μίας επιτυχούς συνεργασίας σε προπτυχιακό επίπεδο για την υλοποίηση της εφαρμογής.

Από τεχνικής άποψης οι στόχοι οι οποίοι προέκυψαν ήταν η υλοποίηση διαδραστικής εφαρμογής τρισδιάστατου περιεχομένου σε φυλλομετρητή χωρίς την χρήση επιπρόσθετου προγράμματος (plug-in). Για να πραγματοποιηθεί αυτό έπρεπε να λυθούν θέματα όπως η μείωση μεγέθους των δεδομένων που αποστέλονται στον χρήστη και η βελτιστοποίηση της απόδοσης της καθώς οι υπολογισμοί για την απεικόνιση καταναλώνουν πολλούς πόρους.

Κατά την διάρκεια της υλοποίησης της εφαρμογής προέκυψε ο πιο βασικός στόχος στην επιστήμη των υπολογιστών, ο οποίος είναι η επεκτασιμότητα της εφαρμογής. Για τον λόγο αυτό υλοποιήσαμε ένα διαδικτυακό εργαλείο διαχείρισης της βασικής εφαρμογής μέσω του οποίου η εφαρμογή μπορεί να αναδείξει περαιτέρω μνημεία της Κρήτης μόλις αυτά σχεδιαστούν. Υλοποιήθηκε επομένως βάση δεδομένων αποθήκευσης των μνημείων καθώς και των πληροφοριών που παρέχει.

7.3 Μελλοντικές επεκτάσεις και βελτιώσεις

Σε κάθε ολοκληρωμένη εφαρμογή πάντα υπάρχουν περιθώρια ανάπτυξης και βελτίωσης της. Το ίδιο συμβαίνει και με την εφαρμογή που υλοποιήθηκε μέσω της διπλωματικής αυτής εργασίας. Μέσω μεθόδων αξιολόγησης της εφαρμογής μπορούν να βρεθούν προβλήματα ευχρηστίας της και να διορθωθούν καθώς και να προκύψουν προτάσεις από τους ίδιους τους χρήστες της αξιολόγησης.

Πιο συγκεκριμένα όσον αφορά την εφαρμογή επισκέπτη, θα μπορούσε να επεκταθεί σε μεγαλύτερο επίπεδο, όπως για παράδειγμα να υπάρξει ένα επίπεδο Greeceπριν το επίπεδο Crete, και ένα Europeακόμη πιο πάνω. Έχοντας σχεδιάσει την βάση δεδομένων με αυτό τον μελλοντικό στόχο, θα μπορούσε να αναπτυχθεί ο όγκος των μνημείων μέσω των επιμέρους επιπέδων λεπτομέρειας. Αντίστοιχα θα μπορούσαν να αυξηθούν και οι Ιστορικές περίοδοι στις οποίες αναφέρεται η εφαρμογή. Επιπλέον μετά τον Ιούλιο του 2014 άρχισε να υποστηρίζεται η χρήση της WebGLσε κινητά τηλέφωνα και ταμπλέτες. Επομένως θα μπορούσε να υλοποιηθεί η αντίστοιχη εφαρμογή για tablets, ενσωματώνοντας όλη την λειτουργικότητα όταν είναι εφικτό από άποψη υλικού(hardware). Παραμένοντας στις ταμπλέτες και τα κινητά θα μπορούσε να εντοπίζει το στίγμα του κινητού και αν βρίσκεται ο χρήστης σε μνημείο να του εμφανίζει το μνημείο που βρίσκεται και πληροφορίες για αυτό. Για τον λόγο

αυτό υλοποιήθηκε η επικοινωνία με API, καθώς έτσι χρειάζεται απλώς η υλοποίηση της γραφικής διεπαφής της ταμπλέτας-κινητού. Επιπλέον με την χρήση των servicesπου παρέχουμε θα μπορούσε να υλοποιηθεί ένα API το οποίο παρέχει συγκεκριμένες τρισδιάστατες σκηνές από μνημεία σε ιστοσελίδες ενδιαφέροντος τους. Τέλος θα μπορούσε να παρέχεται χρήση της εφαρμογής με την χρήση του διαδεδομένου OculusRift (μάσκα εικονικής πραγματικότητας).

Η εφαρμογή διαχείρισης ως υλοποίηση καθαρά προσωπικής ιδέας, παρέχεται και αυτή για βελτιώσεις και μελλοντικές επεκτάσεις. Θα μπορούσε να μετατραπεί εύκολα με την προσθήκη χρηστών σε ένα σύστημα ψηφιακού αποθετηρίου τρισδιάστατων διαδραστικών σκηνών. Ο κάθε χρήστης δηλαδή να μπορεί να υλοποιήσει την δική του τρισδιάστατη εφαρμογή με όσα επίπεδα στον άξονα yκαι x θέλει και εύκολα να δώσει διαδραστικότητα στην εφαρμογή του. Τέλος θα μπορούσε να αυξηθεί ο αριθμός των τύπων τρισδιάστατων μοντέλων που υποστηρίζει η εφαρμογή.

7.4 Επίλογος

Υλοποιώντας την διπλωματική αυτή εργασία μου δόθηκε η ευκαιρία να χρησιμοποιήσω νέες τεχνολογίες για την υλοποίηση διαδικτυακών εφαρμογών, καθώς επίσης και να αναπτύξω την ικανότητα συνεργασίας με μία πολυάριθμη ομάδα. Είχα την ευκαιρία να συμμετέχω σε έργο το οποίο ήταν παραδοτέο, και ως ο μόνος προγραμματιστής απέκτησα εμπειρία για τον τρόπο με τον οποίο υλοποιείται μία εφαρμογή σε συνεργασία με μη προγραμματιστές, κάτι εντελώς πρωτόγνωρο. Επιπλέον η πίεση που ασκήθηκε από τις ημερομηνίες παράδοσης των τριών παραδοτέων, ήταν πολύ σημαντική για την συνέχεια και έδινε δημιουργικό κυρίως χαρακτήρα αντιμετώπισης προβλημάτων τελευταίας στιγμής. Τέλος να τονίσουμε ότι είναι αρκετά πρωτοποριακή μία εφαρμογή αμιγώς τρισδιάστατου χαρακτήρα.

Βιβλιογραφία

[1] Tony Parisi: WebGL: Up and Running. Εκδόσεις: O'Reilly Media(2012)

[2] Diego Cantor, Brandon Jones: WebGL Beginner's Guide. Εκδόσεις: Packt Publishing(2012)

[3] WebGL. https://www.khronos.org/webgl/.

[4] Three.JS. http://threejs.org/.

[5]David Flanagan: JavaScript: The Definitive Guide, 6th Edition. Εκδόσεις: O'Reilly Media (2011)

[6] MySQL. http://www.mysql.com/.

[7] Alan Beaulieu: Learning SQL, 2nd Edition. Εκδόσεις: O'Reilly Media(2009)

[8] Larry Ullman: Visual Quickstart Guide MySQL, 2nd Edition. Εκδόσεις: Peachpit(2008)

[9]Elizabeth Castro: HTML, XHTML, and CSS, Visual QuickStart Guide, 6th Edition. Εκδόσεις: Peachpit(2006)

[10] Mark Pilgrim: Dive Into HTML5, eBook. http://diveintohtml5.info/.

[11] HTML5 : http://www.w3schools.com/html/html5_intro.asp.

[12] CSS: http://www.w3schools.com/css/.

[13] Javascript: http://www.w3schools.com/js/.

[14] HTML5 Reference: http://dev.w3.org/html5/html-author.

[15] Model–view–controller: http://en.wikipedia.org/wiki/Model%E2%80%93View%E2%80%93Controller.

[16] MVC design pattern: http://www.techrepublic.com/article/mvc-design-patternbrings-about-better-organization-and-code-reuse/1049862.

[17] Ajax: http://www.w3schools.com/ajax/

[18] Joshua Eichorn : Understanding AJAX: Using JavaScript to Create Rich Internet Applications.Εκδόσεις:Prentice Hall (2006)

[19] Data Access Object: http://www.oracle.com/technetwork/java/dataaccessobject-138824.html.

[20] Sandeep Kumar Patel: Developing Responsive Web Applications with AJAX and jQuery. Εκδόσεις: Packt Publishing (2014).

[21]Jos Dirksen: Three.js Essentials . Εκδόσεις: Packt Publishing(2014)

[22] Jos Dirksen: Learning Three.js: The javascript 3D library for WebGL. Εκδόσεις: Packt Publishing(2013)

[23] Joydip Kanjilal: ASP.NET Web API: Build RESTful web applications and services on the .NET framework. Εκδόσεις: Packt Publishing(2013)

[24] Fanie Reynders: RESTful Services with ASP.NET Web API. Εκδόσεις: Packt

Publishing(2014)
[25] Alistair Cockburn: Writing Effective Use Cases. Εκδόσεις: Addison-Wesley Professional(2000)
[26] Entity–relationship model: http://en.wikipedia.org/wiki/Entity%E2%80%93relationship_model