Σχολή Τεχνολογικών Εφαρμογών
Τμήμα Εφαρμοσμένης Πληροφορικής & Πολυμέσων

Πτυχιακή εργασία

Τίτλος: Ηλεκτρονικό Βιβλίο

Χρήστος Ζιώγας (ΑΜ: 135)

Επιβλέπων καθηγητής : Νίκος Παπαδάκης

Ευχαριστίες

Ευχαριστώ απο καρδιάς τους ανθρώπους που υποστήριξαν την προσπάθεια μου και ενθάρρυναν, ο καθένας απο τη θέση του, την ολοκλήρωση των σπουδών μου. Ευχαριστώ ιδιαιτέρως τη σύζυγο μου, τους γονείς μου, τους φίλους μου και κυρίως τον κύριο Νίκο Παππαδάκη που μου εμπιστεύτηκε το θέμα της πτυχιακής εργασίας μου, ώστε να πάρω το πτυχίο μου.

Abstract

In the following pages I will analyze the steps to develop an MVC platform and run on top of it a web based ebook. My purpose is to understand the MVC design pattern and the technologies that are necessary for the implementation of this project. The result will be a web application where the user can commit messages and via an administration panel can handle with the messages.

Σύνοψη

Στις σελίδες που ακολουθούν αναλύονται τα βήματα της ανάπτυξης μιας MVC πλατφόρμας πάνω στην οποία θα αναπτυχθεί η εφαρμογή του ηλεκτρονικού βιβλίου. Ο σκοπός είναι η κατανόηση του MVC σχεδιαστικού προτύπου και η τριβή και η ενασχόληση με τις τεχνολογιες που είναι απαραίτητες για την εφαρμογή αυτού του σχεδιαστικού προτύπου. Το αποτέλεσμα θα είναι μια web εφαρμογή με την οποία ο χρήστης θα μπορεί να καταχωρεί μηνύματα και με τη βοήθεια ενός συστήματος διαχείρισης σχολίων να μπορεί να διαχειρίζεται τα μηνύματα αυτά.

Πίνακας περιεχομένων

ΚΕΦΑΛΑΙΟ 1

Εισαγωγή

Στη παρούσα πτυχιακή εργασία περιγράφεται και αναλύεται ο τρόπος καθώς και τα απαράιτητα εργαλεία για την ανάπτυξη και χρήση της εφαρμογής ηλεκτρονικού βιβλίου. Στα επόμενα κεφάλαια θα γίνει εκτένής ανάλυση του σχεδιασμού της πλατφόρμας ανάπτηξης και της ίδιας της εφαρμογής, με την οποία επιτυγχάνεται η καταχώρηση σχολίων και η διαχείρισή τους απο τον εγκεκριμένο χρήστη.

1.1 Περίληψη

Η παρούσα πτυχιακή εργασία πραγματεύται την ανάλυση, σχεδίαση και υλοποίηση, αφενός μιας πλατφόρμας-σκελετού ανάπτυξης (software framework[1]) με βάση το σχεδιαστικό πρότυπο MVC[2] και αφετέρου την ανάπτυξη της εφαρμογής ηλεκτρονικού βιβλίου σύμφωνα με τους κανόνες του MVC.

Για την υλοποίηση του χρησιμοποιούμε έναν web server που φιλοξενεί τόσο την εφαρμογή, μέσω της οποίας γινεται η καταχώρηση σχολίων και η διαχείρισή τους απο εγκεκριμένο χρήστη, όσο και τη βάση δεδομένων με την οποία αντλεί και αποθηκεύει τα δεδομένα των σχολίων. Οι τεχνολογίες και τα εργαλεία που χρησιμοποιήθηκαν είναι κυρίως ανοιχτού κώδικα και δωρεάν για χρήση και ανάπτυξη εφαρμογών. Ο web server ο οποίος χρησιμοποιήθηκε ειναι ο Apache[3] σε συνδυασμό με βάση δεδομένων τύπου MySQL[4] και για την ανάπτυξη πηγαίου κώδικα γλωσσες προγραμματισμού HTML[5], CSS[6], PHP[7], Javascript[8] και JQuery[9].

Το σύστημα θα δίνει τη δυνατότητα σε απλούς χρήστες να καταχωρούν χωρίς εγγραφή το σχόλιό τους μέσω διαδικτύου αλλά και στον διαχειριστή του να μπορεί να αποδέχεται και να εγκρίνει ή να απορρίπτει σχόλια των χρηστών. Επίσης, θα δίνει την δυνατότητα στον προγραμματιστη την περαιτέρω ανάπτυξη feature[10] της εφαρμογής μέσα σε ένα δομημένο πλαίσιο και κανόνων ανάπτυξης προγραμμάτισμού κοινώς αποδεκτό.

---

[1] Software Framework https://en.wikipedia.org/wiki/Software_framework

[2] MVC https://el.wikipedia.org/wiki/Model-view-controller

[3] Apache Software Fountation http://www.apache.org/

[4] MySQL https://www.mysql.com/

[5] HTML https://el.wikipedia.org/wiki/HTML

[6] CSS https://en.wikipedia.org/wiki/Cascading_Style_Sheets

[7] PHP https://secure.php.net/

[8] Javascript https://www.javascript.com/

[9] jQuery https://jquery.com/

[10] software feature https://en.wikipedia.org/wiki/Software_feature

1.2 Κίνητρο για τη διεξαγωγή της εργασίας

Με τη ραγδαία ανάπτυξη των web υπηρεσιών γεννώνται ολοένα και περισσότερες νέες ανάγκες και απαιτήσεις, που αποζητούν την ικανοποίησή τους μέσα από πιο εξελιγμένα πρότυπα και νέες μεθόδους προγραμματισμού. Αυτό μας οδήγησε στα web frameworks, τα οποία είναι επαναχρησιμοποιήσιμο προγραμματιστικό περιβάλλον που παρέχει πολλαπλές λειτουργίες, κάθε μια από τις οποίες αποτελούν μέρος μιας μεγαλύτερης καθολικής πλατφόρμας προγραμματισμού, ώστε να διευκολύνουν την ανάπτυξη του λογισμικού των εφαρμογών, των προϊόντων και των λύσεων τους.

Αυτό το φαινόμενο δημιούργησε νέες θέσεις εργασίας βασισμένες στη γνώση των web frameworks και μου έδωσε το κίνητρο να αναπτύξω ένα προσωπικό web framework βασισμένο στις μεθόδους που αξιοποιούν διάσημα web frameworks με σκοπό τη βαθιά κατανόησή τους, ώστε να είμαι σε θέση να ανταπεξέλθω στις ανάγκες της αγοράς.

Η ανάπτυξη του ηλεκτρονικού βιβλίου σε αυτή την MVC πλατφόρμα θα αποτελέσει ένα είδους proof of concept[11] στο πώς να αξιοποιεί κανείς ένα web framework στην υλοποίηση μιας οποιασδήποτε web εφαρμογής.

1.3 Σκοπός και στόχοι εργασίας

Σκοπός της πτυχιακής εργασίας είναι η περαιτέρω εξέλιξη των γνώσεων μου στις βάσεις δεδομένων,την ανάπτυξη web εφαρμογών και των γλωσσών προγραμματισμού (HTML, CSS, PHP, Javascript), στα πλαίσια του σχεδιαστικού αρχιτεκτονικού προτύπου MVC.

Σημαντικοί στόχοι για την εκπόνηση της εργασίας.

1. Σχεδιασμός αρχιτεκτονικής του web framework με βάση το MVC
2. Ανάπτυξη των βασικών στοιχείων του MVC με χρήση PHP
3. Ανάλυση λειτουργικών απαιτήσεων της εφαρμογης ηλεκτρονικού βιβλίου
4. Σχεδιασμός και υλοποιήση διεπαφών της εφαρμογής του ηλεκτρονικού βιβλίου
5. Σχεδιασμός και ανάπτυξη Βάσης δεδομένων με χρήση MySQL
6. Ανάπτυξη και υλοποιήση με χρήση PHP και javascript των λειτουργικών απαιτήσεων του ηλεκτρονικού βιβλίου στα πλάισια του web framework
7. Έλεγχος λειτουργίας και διόρθωση σφαλμάτων
8. Συγγραφή αναφοράς εργασίας
9. Υποβολή αίτησης αξιολόγησης εργασίας
10. Παρουσίαση αναφοράς

1.4 Δομή Εργασίας

[11] Proof of concept https://en.wikipedia.org/wiki/Proof_of_concept

ΚΕΦΑΛΑΙΟ 2: Μεθοδολογία υλοποίησης

2.1 Μέθοδος Ανάλυσης & Ανάπτυξης Πτυχιακής

Η ανάπτυξη ενός πληροφοριακού συστήματος για τη διαχείριση ενός ηλεκτρονικού βιβλίου είναι μια web εφαρμογή στην οποία πραγματοποιείται καταχώρηση σχολίων, με τον όρο καταχώρηση αυτόματως σημαίνει οτι απαιτηται η χρήση βάσης δεδομένων με την οποία θα αντλεί και θα αποθηκεύει τα δεδομένα των σχολίων.

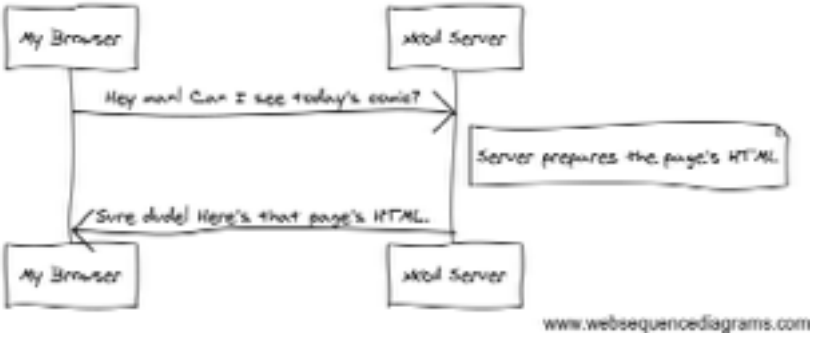
2.2 Θεωρίες

2.2.1 HTTP

HTTP συντομογραφία της φράσης: «Hypertext Transfer Protocol». Είναι ένα σύνολο κανόνων, ή αλλιώς πρωτόκολλο, που καθορίζει τον τρόπο με τον οποίο θα γίνει η μεταφορά του υπερκειμένου (hypertext) μεταξύ δύο ή περισσοτέρων υπολογιστών.

Το πρωτόκολλο HTTP είναι το πιο συνηθισμένο στον ηλεκτρονικό χώρο του World Wide Web. Η ονομασία του προέρχεται από τα αρχικά των αγγλικών λέξεων HyperText Transfer Protocol (Πρωτόκολλο Μεταφοράς Υπερκειμένου). Το πρωτόκολλο αυτό χρησιμοποιε ίται από τη συγκεκριμένη υπηρεσία του δικτύου Internet από το 1990. Το HTTP αποτελεί ένα πρωτόκολλο του επιπέδου εφαρμογών στα δίκτυα υπολογιστών και κυρίως σε διανεμημένα πληροφορικά συστήματα υπερμέσων. Είναι ένα γενικό, αντικειμενοστραφές πρωτόκολλο που μπορεί να χρησιμοποιηθεί σε ένα πλήθος εφαρμογών, για παράδειγμα σε εξυπηρετητές-διανομείς (servers) και διανεμημένα συστήματα διαχ ίρισης αντικειμένων. Το βασικότερο και πιο σημαντικό ίσως χαρακτηριστικό του πρωτοκόλλου αυτού είναι ότι επιτρέπει στα διάφορα συστήματα μετάδοσης δεδομένων να υφίστανται ανεξάρτητα από τα δεδομένα που αυτά μεταφέρουν. Το πρωτόκολλο που χρησιμοποιείται για τη μεταφορά των δεδομένων και των κλήσεων από τον Web server στον Web browser (και αντίστροφα) ονομάζεται HTTP (HyperText Transfer Protocol).

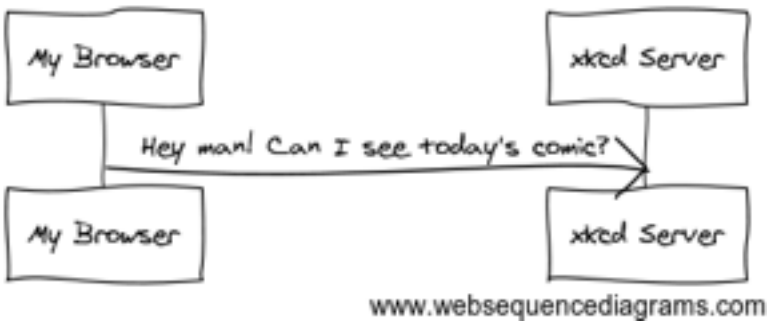
Το HTTP με λίγα λόγια είναι μια text γλώσσα με την οποία επιτρέπει δύο μηχανήματα να επικοινωνούν μεταξύ τους. Για παράδειγμα στην Εικόνα 1. παρουσιάζεται αυτη η επικοινωνία, παρακάτω θα παρουσιάστεί σε 2 βήματα πως λειτουργει το πρωτοκόλλο.



Εικόνα 1

2.2.1.1 Ο Client στέλνει Request

Κάθε συνεδρία στο web αρχίζει με ένα request. Το request είναι ένα text μηνυμα που έχει δημιουργηθεί απο τον client (πχ ένα browser, μία iPhone εφαρμογή, κλπ), σε μια δομημένη μορφή κειμένου γνωστή ως HTTP. Ο client στέλνει request στον server και έπειτα περιμένει απάντηση.



Εικονα 2

Σε γλώσσα HTTP, αυτό το HTTP request στην πραγματικότητα θα ήταν η παρακάτω εικόνα.

```
1  GET / HTTP/1.1
2  Host: xkcd.com
3  Accept: text/html
4  User-Agent: Mozilla/5.0 (Macintosh)
```

Εικόνα 3

Στην Εικόνα 3 το μήνυμα αναφέρει όλες τις απαραίτητες πληροφορίες για το ποιο resource ζητά(request) ο client. Η πρώτη γραμμη είναι η πιο σημαντική και περιέχει 2 πράγματα, το URI και την HTTP μέθοδο. Το URI (πχ, /, /contact, κλπ) είναι μια μοναδική διεύθυνση ή τοποθεσία η οποία αναγνωριζει τον web πόρο (resource) που θέλει ο client. Η HTTP μέθοδος (πχ, GET) ορίζει το τι θελουμε να κάνουμε με το resource. Οι HTTP μέθοδοι (Εικόνα 3) είναι τα ρήματα (ενέργειες) του request και ορίζουν συγκεκριμένους τρόπους με τους οποίους μπορούμε να χρησιμοποιήσουμε για το resource.

GET	Retrieve the resource from the server
POST	Create a resource on the server
PUT	Update the resource on the server
DELETE	Delete the resource from the server

Εικόνα 4

Υπάρχουν 9 συνολικά μέθοδοι, αλλά πολλοί από αυτές είτε δεν χρησιμοποιούνται συχνά είτε δεν υποστηρίζονται από τους σύγχρονους Browsers. Εκτός από την πρώτη γραμμή, ένα αίτημα HTTP περιέχει πάντοτε άλλες γραμμές των πληροφοριών που ονομάζεται HTTP header request (κεφαλίδες αίτησης). Οι HTTP headers μπορεί να παρέχουν ένα ευρύ φάσμα πληροφοριών, όπως ο ο host , τα response formats που δέχεται ο Client (Accept) και πληροφορίες όπως με τη Client εφαρμογή με την οποία έγινε το request (User-Agent).

2.2.1.2 Ο Server επιστρέφει Response

Όταν ο διακομιστής δεχθεί ένα request, γνωρίζει ακριβώς ποια resource θέλει ο client (μέσω URI) και τι θέλει να κάνει με αυτή (μέσω HTTP method). Για παράδειγμα, στη περίπτωση ενός GET request ο διακομιστής ετοιμάζει το resource και επιστρέφει ένα HTTP response, η Εικόνα 1 περιγράφει ακριβως αυτη την ενέργεια.

Σε γλωσσα HTTP, η απάντηση (response) που επιστρέφεται στον browser θα έδειχνε όπως στην Εικόνα 5

```
1  HTTP/1.1 200 OK
2  Date: Sat, 02 Apr 2011 21:05:05 GMT
3  Server: lighttpd/1.4.19
4  Content-Type: text/html
5
6  <html>
7    <!-- ... HTML for the xkcd comic -->
8  </html>
```

Εικόνα 5

Το HTTP response περιέχει το ζητούμενο resource (HTML στην περίπτωση μας), μαζί και με άλλες πληροφορίες για αυτο. Η πρώτη γραμμή είναι πολύ σημαντική αναφέρει τον HTTP status code (200 στη περίπτωση μας). Το status code αναφέρει το συνολικό αποτέλεσμα του request πίσω στον διακοσμιτή, ποιο συγκεκριμένα αναφέρει αν υπήρξε κάποιο σφαλμα, ή για το άν ο client χρειάζεται να κάνει κάτι (πχ, ανακατεύθυνση σε άλλη σελίδα). Όπως το Request ετσι και το HTTP response περιέχει επιπρόσθετες πληροφορίες γνωστές ώς HTTP headers (HTTP Κεφαλίδες). Για παράδειγμα μια πολυ

σημαντική Κεφαλίδα είναι το Content-type. Η κύρια πληροφορία θα μπορούσε να επιστραφεί σε πολλαπλά διαφορετικά formats όπως HTML, XML ή JSON, εδω πέρνει θέση η Content-type header, ή οποία χρησιμοποιεί Internet media types όπως το text/html για να πει στον client ποιο format επιστρέφεται.

2.2.1.3 Request, Response και προγραμματισμός διαδικτύου

Αυτή η request-response επικοινωνία ειναι η θεμελιώδης διεργασία που οδηγεί όλες τις επικονωνίες στο διαδίκτυο και όσο σημαντική και ισχυρή είίναι αυτη η διαδικασία τόσο θαυμάσια απλή είναι. Το σημαντικότερο γεγονός ίσως είναι οτι ασχέτως ποια γλωσσα προγραμματισμού χρησιμοποιούμε ή τον τύπο της εφαρμογής που αναπτύσουμε ο βασικός στόχος είναι πάντα να αναγνωρίζουμε το κάθε request και να δημιουργούμε και να στέλνουμε το αντίστοιχο response. Στη παρούσα εργασία η server side γλωσσα προγραμματισμού είναι η PHP, στην Εικόνα 6 θα δέιτε ένα απλο παράδειγμα υλοποιησης της request-response επικοινωνίας σε PHP.

```php
$uri = $_SERVER['REQUEST_URI'];
$foo = $_GET['foo'];

header('Content-type: text/html');
echo 'Το URI που ζητήθηκε είναι: '.$uri;
echo 'Η τιμή της "foo" παράμετρου είναι: '.$foo;
```

Εικόνα 6

Αυτή η μικρή εφαρμογή πάιρνει πληροφορίες απο το HTTP request και το χρησιμοποιεί για να δημιουργήσει ένα HTTP response. Εδώ με PHP υπερ μεταβλητές όπως ή $_SERVER και η $_GET πέρνουμε την request πληροφορία, και με την header() συνάρτηση δημιουργούμε response headers και μαζι με αυτο περιεχόμενο της πληροφορίας. Η PHP θα δημιουργήσει HTTP response και θα επιστεψει την πληροφορία στον Client.

```
HTTP/1.1 200 OK
Date: Sat, 03 Jul 2014 02:14:33 GMT
Server: Apache/2.2.17 (Unix)
Content-Type: text/html

Το URI που ζητήθηκε είναι: /testing?foo=christos
Η τιμή της "foo" παράμετρου είναι: christos
```

Εικόνα 7

2.2.2 Web Document

Το Web document έχει την ίδια σημασιολογία με την Ιστοσελίδα, μόνο που αυτή η ονομασία ικανοποιεί τον ορισμό του W3C. Κάθε web document έχει την δική του URI, πρέπει να τονιστεί οτι ένα web document δεν αντιστοιχεί απαραίτητα σε ένα αρχείο, αυτο σημαίνει οτι ένα μοναδικό web document μπορεί να είναι διαφορετικά format και γλωσσες προγραμματισμού, όπως επίσης κ ένα μοναδικό αρχείο. Για παράδειγμα ένα php script μπορεί να ειναι υπεύθυνο για τη δημιουργία πολλαπλών web documents με διαφορετικά URI. Το web document ορίζεται ως κάτι που έχει URI και μπορεί να επιστρέφει representations (σε μορφες όπως HTML ή JPEG ή RDF) των πόρων (resources) που προσδιορίζονται ως απάντηση (response) σε αιτήματα HTTP (HTTP requests).

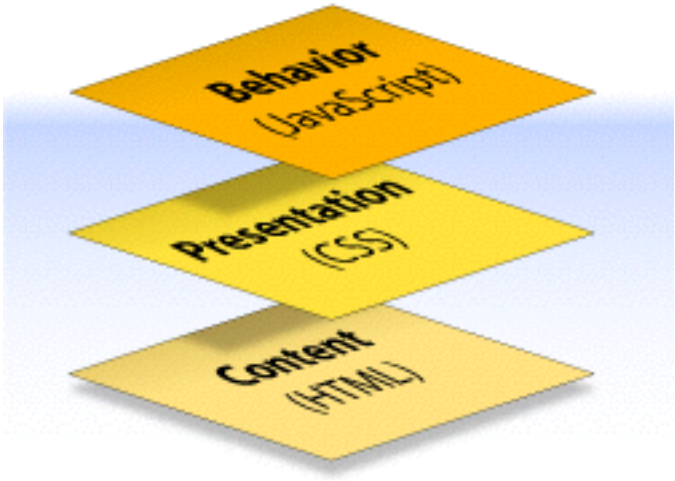
2.2.2.1 Τα επίπεδα του Web Document

Ένα Web document μπορεί να αποτελείται μέχρι 3 επιπεδα(layers)-περιεχόμενο(Content or Structure), παρουσίαση(Style or Presentation) κ συμπεριφορά(Behavior), τα οποια θα τα αναφέρουμε στην επόμενη υποενότητα. Στον προγραμματισμό διαδικτύου είναι σημαντικό να διαχωρίζουμε αυτά τα επίπεδα για να αποκτήσουμε τα παρακάτω πλεονεκτήματα.

Κοινόχρηστα resources: Οταν γράφεις ενα εξωτερικό css αρχείο ή javascript, σου δινεται η δυνατότητα να χρησιμοποιούνται αυτα τα αρχεία απο οποιαδήποτε ιστοσελίδα στον ιστοτοπο. Με αυτό γλυτώνεις διπλό κοπο όταν αυτα τα αρχεία χρειαζονται επεξεργασία και οι αλλαγές θα ισχύουν για κάθε web document τα χρησιμοποιεί.

Ταχύτερη λήψη(downloads): Αρκεί μια φορά να κατέβει το css αρχείο ή javascript απο τον client διοτι το αρχείο γινεται αποθηκεύεται προσωρινά απο τον Web browser με αποτελεσμα η ιστοσελίδα να φορτώσει γρηγορότερα στον browser.

Δυνατοτητα ομάδα εργασίας πολλαπλών ατόμων: Αν υπάρχει ταυτοχρονη υλοποιηση πάνω απο έναν προγραμματιστή σε ένα έργο δινεται η δυνατότητα να μοιραστεί ο φορτος εργασίας χωρις να ανησυχουμε για δικαιώματα και διαχείρηση περιεχόμενου. Μπορουμε να αναθεσουμε για παραδειγμα στους web designers τον φορτο εργασίας του design και ταυτοχρονα οι javascript προγραμματιστες να υλοποιουν το behavior μερος του έργου.



Εικόνα 8

2.2.1.1 HTML και Επίπεδο Περιεχομένου

Το Επίπεδο περιεχομένου (content layer), είναι πάντα παρών, περιλαμβάνει τις πληροφορίες που θέλουμε να μεταδωσουμε στον client -user και εντασεται στο πλαίσιο HTML σημανσης. Ενα απο τα μεγαλύτερα μέρη του περιεχόμενου έιναι το κείμενο, αλλα μπορεί να επίσης να παρέχεται μέσα απο εικόνες και βιντεο και οτιδηποτε αλλο θελουμε να παρουσιασουμε.

Η HTML (ακρωνύμιο του αγγλικού Hypertext Markup Language, ελλ. Γλώσσα Σήμανσης Υπερκειμένου) είναι η κύρια γλώσσα σήμανσης για τις ιστοσελίδες, και τα στοιχεία της είναι τα βασικά δομικά στοιχεία των ιστοσελίδων. Η HTML γράφεται υπό μορφή στοιχείων HTML τα οποία αποτελούνται από ετικέτες (tags), οι οποίες περικλείονται μέσα σε σύμβολα «μεγαλύτερο από» και «μικρότερο από» (για παράδειγμα <html>), μέσα στο περιεχόμενο της ιστοσελίδας. Οι ετικέτες HTML συνήθως λειτουργούν ανά ζεύγη (για παράδειγμα <h1> και </h1>), με την πρώτη να ονομάζεται ετικέτα έναρξης και τη δεύτερη ετικέτα λήξης (ή σε άλλες περιπτώσεις ετικέτα ανοίγματος και ετικέτα κλεισίματος αντίστοιχα). Ανάμεσα στις ετικέτες, οι σχεδιαστές ιστοσελίδων μπορούν να τοποθετήσουν κείμενο, πίνακες, εικόνες κλπ.

Ο σκοπός ενός web browser είναι να διαβάζει τα έγγραφα HTML και τα συνθέτει σε σελίδες που μπορεί κανείς να διαβάσει ή να ακούσει. Ο browser δεν εμφανίζει τις ετικέτες HTML, αλλά τις

Event	Description	DOM
onclick	The event occurs when the user clicks on an element	2
oncontextmenu	The event occurs when the user right-clicks on an element to open a context menu	3
ondblclick	The event occurs when the user double-clicks on an element	2
onmousedown	The event occurs when the user presses a mouse button over an element	2
onmouseenter	The event occurs when the pointer is moved onto an element	2
onmouseleave	The event occurs when the pointer is moved out of an element	2
onmousemove	The event occurs when the pointer is moving while it is over an element	2
onmouseover	The event occurs when the pointer is moved onto an element, or onto one of its children	2
onmouseout	The event occurs when a user moves the mouse pointer out of an element, or out of one of its children	2
onmouseup	The event occurs when a user releases a mouse button over an element	2

Εικόνα 9

χρησιμοποιεί για να ερμηνεύσει το περιεχόμενο της σελίδας.

Τα στοιχεία της HTML χρησιμοποιούνται για να κτίσουν όλους του ιστότοπους. Η HTML επιτρέπει την ενσωμάτωση εικόνων και άλλων αντικειμένων μέσα στη σελίδα, και μπορεί να χρησιμοποιηθεί για να εμφανίσει διαδραστικές φόρμες. Παρέχει τις μεθόδους δημιουργίας δομημένων εγγράφων (δηλαδή εγγράφων που αποτελούνται από το περιεχόμενο που μεταφέρουν και από τον κώδικα μορφοποίησης του περιεχομένου) καθορίζοντας δομικά σημαντικά στοιχεία για το κείμενο, όπως κεφαλίδες, παραγράφους, λίστες, συνδέσμους, παραθέσεις και άλλα. Μπορούν επίσης να ενσωματώνονται σενάρια εντολών σε γλώσσες όπως η JavaScript, τα οποία επηρεάζουν τη συμπεριφορά των ιστοσελίδων HTML.

Οι Web browsers μπορούν επίσης να αναφέρονται σε στυλ μορφοποίησης CSS για να ορίζουν την εμφάνιση και τη διάταξη του κειμένου και του υπόλοιπου υλικού. Ο οργανισμός W3C, ο οποίος δημιουργεί και συντηρεί τα πρότυπα για την HTML και τα CSS, ενθαρρύνει τη χρήση CSS αντί διαφόρων στοιχείων της HTML για σκοπούς παρουσίασης του περιεχομένου.

2.2.1.2 Επίπεδο παρουσίασης

Το επίπεδο παρουσίασης (Style Layer) καθορίζει το πως θα εμφανίζεται το περιεχόμενο σε έναν χρήστη που έχει πρόσβαση στο web document. Ο συμβατικός τρόπος για να δει κάποιος μια σελίδα είναι με ένα συνηθισμένο πρόγραμμα περιήγησης φυσικά, αλλά αυτός είναι ένας απο τους πιθανούς τρόπους πρόσβασης. Για παράδειγμα, το περιεχόμενο θα μπορούσε να μετατραπεί σε ομιλία για χρήστες που παρουσιάζουν προβλήματα όρασης ή αναγνωσης.

Το επίπεδο παρουσίασης ουσιαστικά ειναι τόσο μια φιλοσοφία σχεδιασμού οσό και μεθοδολογία ή οποία εφαρμόζεται στα πλαισιο διαφορων τεχνολογικών αρχών, συμπεριλαμβανομένων της ανάκτησης της πληροφορίας, επεξεργασίας προτύπων, επεξεργασία κειμένων, προγραμματισμός διαδικτύου κ.α. Αποτελεί μέρος ενός γενικότερου σχεδιαστικού προτύπου το separation of concerns (SoC)[12], που αποσκοπεί στον διαχωρισμό της παρουσίασης ενός document απο το περιεχόμενό του.

Αυτός ο διαχωρισμός μορφοποίησης και περιεχομένου μας καθιστά δυνατό να παρουσιάσουμε την ίδια HTML markup[13] σε διαφορετικά στυλ, όπως για εκτύπωση, για ομιλία κλπ. Μπορεί επίσης να χρησιμοποιηθεί στο να δείξει μια ιστοσελίδα σε διαφορετικές εκδοχές ανάλογα με το μέγεθος της οθόνης.

2.2.1.3 Javascript και Επίπεδο Συμπεριφοράς

Το επίπεδο συμπεριφοράς (Behavior Layer) περιλαμβάνει την αλληλεπίδραση του χρήστη σε πραγματικό χρόνο με το web document. Εδώ παίρνει θέση το Document Object Model[14] συχνά αναφερόμενο ως DOM.

Το DOM είναι ενα ανεξαρτήτως πλατφόρμας και γλώσσας προγραμματισμού λογισμικό που αναλαμβάνει την αναπαράσταση και την αλληλεπίδραση με τα objects[15] των HTML, XHTML, και XML web documents. Παρουσιάστηκε απο την Netscape Comunications το 1995 με την έκδοση του Netscape Navigator 2 ως port[16] της Javascript. Τα DOM Events επιτρέπουν στη javascript να καταγράψουν τους event handlers μεσα σε ενα web document. Τα events[17] συνήθως χρησιμοποιούνται σε συνδυασμό με υπορουτίνες οι οποίες δεν εκτελούνται πρωτου συμβεί κάποιο event (πχ. όταν ο χρηστης κάνει click ένα κουμπί). Ενα μικρό δειγμα βλεπουμε στην εικόνα 9 που αναφέρονται τα mouse events.

---

12 https://en.wikipedia.org/wiki/Separation_of_concerns

13 https://developer.mozilla.org/en-US/docs/Web/HTML

14 https://en.wikipedia.org/wiki/Document_Object_Model

15 https://en.wikipedia.org/wiki/Object_(computer_science)

16 https://en.wikipedia.org/wiki/Porting

17 https://developer.mozilla.org/en-US/docs/Web/API/Event

2.2.3 Πλατφόρμα εφαρμογών Web (Web application framework)

Το web application framework (WAF)[18] ή web πλατφόρμα εργασίας είναι software framework τα οποία είναι ειδικά σχεδιασμένα στο να υποστηριζουν την ανάπτυξη ιστοσελίδων, web εφαρμογών, web υπηρεσιών και web πόρων. Η πλατφόρμες αυτές αποσκοπούν στο να ανακουφίσει την επιβάρυνση στους προγραμματιστές στην ανάπτυξη web εφαρμογών. Για παράδειγμα, πολλά frameworks διαθέτουν βιβλιοθήκες για προσβαση σε βάσεις δεδομένων, templating[19] frameworks και διαχείριση συνόδων (sessions)[20], που ενθαρύνουν την επαναχρησιμοποίηση κώδικα[21].

[18] https://en.wikipedia.org/wiki/Web_application_framework

[19] https://en.wikipedia.org/wiki/Template_processor

[20] https://en.wikipedia.org/wiki/Session_(computer_science)

[21] Multiple (wiki). "Web application framework". Docforge. Retrieved 2010-01-19.

ΚΕΦΑΛΑΙΟ 3 Εργαλεία ανάπτυξης εργασίας

3.1 HTML

Η HTML γράφεται υπό μορφή στοιχείων HTML τα οποία αποτελούνται από ετικέτες (tags), οι οποίες περικλείονται μέσα σε σύμβολα «μεγαλύτερο από» και «μικρότερο από» (για παράδειγμα <html>), μέσα στο περιεχόμενο της ιστοσελίδας. Οι ετικέτες HTML συνήθως λειτουργούν ανά ζεύγη (για παράδειγμα <h1> και </h1>), με την πρώτη να ονομάζεται ετικέτα έναρξης και τη δεύτερη ετικέτα λήξης (ή σε άλλες περιπτώσεις ετικέτα ανοίγματος και ετικέτα κλεισίματος αντίστοιχα). Ανάμεσα στις ετικέτες, οι σχεδιαστές ιστοσελίδων μπορούν να τοποθετήσουν κείμενο, πίνακες, εικόνες κλπ.

Ο σκοπός ενός web browser είναι να διαβάζει τα έγγραφα HTML και τα συνθέτει σε σελίδες που μπορεί κανείς να διαβάσει ή να ακούσει. Ο browser δεν εμφανίζει τις ετικέτες HTML, αλλά τις χρησιμοποιεί για να ερμηνεύσει το περιεχόμενο της σελίδας.

Τα στοιχεία της HTML χρησιμοποιούνται για να κτίσουν όλους του ιστότοπους. Η HTML επιτρέπει την ενσωμάτωση εικόνων και άλλων αντικειμένων μέσα στη σελίδα, και μπορεί να χρησιμοποιηθεί για να εμφανίσει διαδραστικές φόρμες. Παρέχει τις μεθόδους δημιουργίας δομημένων εγγράφων (δηλαδή εγγράφων που αποτελούνται από το περιεχόμενο που μεταφέρουν και από τον κώδικα μορφοποίησης του περιεχομένου) καθορίζοντας δομικά σημαντικά στοιχεία για το κείμενο, όπως κεφαλίδες, παραγράφους, λίστες, συνδέσμους, παραθέσεις και άλλα. Μπορούν επίσης να ενσωματώνονται σενάρια εντολών σε γλώσσες όπως η JavaScript, τα οποία επηρεάζουν τη συμπεριφορά των ιστοσελίδων HTML.

Οι Web browsers μπορούν επίσης να αναφέρονται σε στυλ μορφοποίησης CSS για να ορίζουν την εμφάνιση και τη διάταξη του κειμένου και του υπόλοιπου υλικού. Ο οργανισμός W3C, ο οποίος δημιουργεί και συντηρεί τα πρότυπα για την HTML και τα CSS, ενθαρρύνει τη χρήση των CSS αντί διαφόρων στοιχείων της HTML για σκοπούς παρουσίασης του περιεχομένου.

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>

<h1>My First Heading</h1>
<p>My first paragraph.</p>

</body>
</html>
```

Εικόνα 10

3.1.2 HTML5

Περιλαμβάνει την 5η αναθεώρηση της HTML, CSS3 και μια σειρά από javaScript APIs. Όλες αυτές οι τεχνολογίες μαζί δίνουν την δυνατότητα να δημιουργεί κανείς περίπλοκες εφαρμογές που προηγουμένως θα μπορούσαν να δημιουργηθούν μόνο για desktop platforms.

Δεν ανήκει σε κάποια συγκεκριμένη εταιρία ή σε κάποιον browser. Έχει δημιουργηθεί από κοινότητα ανθρώπων που ενδιαφέρονται για την ανάπτυξη του διαδικτύου και από μια σύμπραξη μεγάλων εταιριών τεχνολογίας όπως Google, Microsoft, Apple, Mozilla, Facebook, IBM, HP, Adobekαι πολλές άλλες. Η κοινότητα των ανθρώπων αυτών και η κοινοπραξία των εταιριών συνεχίζουν να συνεργάζονται πάνω στα ενιαία πρότυπα των browser για να διευρύνουν τις δυνατότητές τους ακόμα περισσότερο. Η επόμενη γενιά των web εφαρμογών μπορούν να τρέξουν υψηλών επιδόσεων γραφικά, εργασία χωρίς σύνδεση, αποθήκευση μεγάλου όγκου δεδομένων στον client, γρήγορη εκτέλεση υπολογισμών και να πάνε την διαδραστικότητα και την συνεργασία ακόμα πιο μακριά. Παρακάτω θα γίνει μια σύνοψη των νεων χαρακτηριστικών που φέρνει η τεχνολογία αυτή.

Πολυμέσα και γραφικά: Οι προγραμματιστές που θέλουν αν δημιουργήσουν συναρπαστικά παιχνίδια, γρήγορα κινούμενα σχέδια, ή εξελιγμένα οπτικά εφέ, είτε έπρεπε να στραφούν σε μια διαφορετική πλατφόρμα, είτε να προβούν στην χρήση plugins. Με την HTML5 ο browser έχει γίνει μια πραγματικά ολοκληρωμένη πλατφόρμα για παιχνίδια, κινούμενα σχέδια, ταινίες και γενικά οτιδήποτε γραφικό. Στοιχεία όπως ο φωτισμός και οι σκιές, οι αντανακλάσεις και η πλούσια υφή των αντικειμένων έχουν σαν αποτέλεσμα ρεαλιστικές συνθέσεις. Υψηλής απόδοσης χαρακτηριστικά όπως 3D CSS, διανυσματικά γραφικά και WebGL turbocharge εφαρμογές με υπέροχα 3D γραφικά και special effects. Πλούσια APIs ήχου και μικρού χρόνου αναμονής δικτύωσης των WebSockets μαζί με τα APIs των γραφικών επιτρέπουν να δημιουργήσετε μια συναρπαστική εμπειρία για τους χρήστες και το κοινό.
Φυσικά καμία από τις παραπάνω τεχνολογίες δεν θα ήταν χρήσιμη αν δεν ήταν σε θέση να λειτουργήσει. Όμως οι μηχανές javaScript έχουν γίνει αρκετά γρήγορες για να τρέξουν αυτά τα γραφικά και να διαχειρίζονται video σε πραγματικό χρόνο. Επίσης οι νέοι browsers τώρα χρησιμοποιούν την μονάδα επεξεργασίας γραφικών (GPU) για να επιταχύνουν τους υπολογισμούς που χρειάζονται για την ομαλή μετάβαση, μετατροπή και το 3D rendering.

Εκτός σύνδεσης και αποθήκευση: Οι έννοιες web και offline είναι κάτι που πολλοί άνθρωποι δεν μπορούν νασυνδυάσουν. Αλλά σύντομα θα το κάνουν καθώς τα APIs της HTML5 θα δίνουν την δυνατότητα να δημιουργούμε εφαρμογές που θα δουλεύουν ακόμα και όταν είμαστε αποσυνδεδεμένοι από το δίκτυο. Η μνήμη cache, η τοπική αποθήκευση, IndexedDB, Σύστημα αρχείων και γεγονότα σύνδεσης - αποσύνδεσης καθιστούν ικανές τις εφαρμογές να δουλεύουν με ή χωρίς σύνδεση. Οι χρήστες επίσης μπορούν να κατεβάζουν μεγάλα αρχεία, μεγαλύτερα από 1GB πλήρη ή μέρος τους για προβολή χωρίς σύνδεση αργότερα. Πέρα από το γεγονός ότι οι εφαρμογές μπορούν να διατηρήσουν την κατάστασή τους και να κρατήσουν δεδομένα χωρίς να υπάρχει σύνδεση με τον server τα features εκτός σύνδεσης έχουν το πλεονέκτημα ότι βελτιώνουν την απόδοση μιας εφαρμογής με το να αποθηκεύουν τα δεδομένα στην cache ή με το να κάνουν τα δεδομένα ακριβή ανάμεσα στα sessions των user και την επαναφόρτωση των σελίδων.

Απόδοση: Η HTML5 επιτρέπει σε web εφαρμογές να ανταποκρίνονται καλύτερα, έτσι
ώστε να δημιουργεί μια εμπειρία στον χρήστη που συναγωνίζεται την εμπειρία των desktop εφαρμογών. Τα APIs εκτός σύνδεσης δεν εξυπηρετούν μόνο για την αποθήκευση αρχείων τοπικά, αλλά επιπροσθέτως βελτιώνουν την απόδοση. Επιτρέπουν στην εφαρμογή γρήγορη πρόσβαση στα τοπικά αποθηκευμένα

δεδομένα και μειώνει τον κύκλο των ερωτήσεων/αιτημάτων που χρειάζεται να κάνει στον server. Μπορεί να κρατήσει στην cache σελίδες που χρησιμοποιούν

περισσότερο οι χρήστες και να αποθηκεύσει το ιστορικό που χρειάζεται για το επόμενο στάδιο. Έχει ως αποτέλεσμα ταχύτερη φόρτωση. Οι μηχανές της javaScript έχουν εξελιχθεί, είναι άριστα βελτιστοποιημένες ώστε να τρέχουν την javaScript γρήγορα. Πέρα από τις νέες αυτές τεχνολογίες μια ποικιλία από τεχνικές μπορούν να ελαχιστοποιήσουν το μέγεθος των εφαρμογών, όπως η ελαχιστοποίηση χρήσης του εύρους ζώνης και των συνδέσεων με τον server, συμπίεση αρχείων. Επίσης, υπάρχει η δυνατότητα πρόσβασης σε βελτιωμένης απόδοσης βιβλιοθήκες και εργαλεία.

Εύκολος προγραμματισμός: Κατ 'αρχάς, η HTML5 επιτρέπει να στοχεύσουμε το μεγαλύτερο αριθμό συσκευών με την λιγότερη προγραμματιστική προσπάθεια. Δεύτερον, σύγχρονα προγράμματα περιήγησης και διάφορες τεχνικές έχουν ελαχιστοποιήσει το fragmentation. Τέλος, HTML5 είναι πιο προσιτή σε ένα ευρύτερο φάσμα προγραμματιστών.

Broad Reach: Μπορούμε να συνδεθούμε με χρήστες οπουδήποτε κι αν βρισκόμαστε με την χρήση HTML5 εφαρμογών που μπορούν να αναπτυχθούν σε πολλές πλατφόρμες και σε μια ευρεία γκάμα συσκευών. Είτε πρόκειται για εταιρία, δημιουργία παιχνιδιών, ή προσωπικών εφαρμογών, η HTML5 επιτρέπει στους χρήστες να έχουν γρήγορη πρόσβαση σε βασικές εφαρμογές. Η HTML5 υποστηρίζεται από όλους τους σύγχρονους browsers και τις βασικότερες συσκευές τηλεφώνων. Καμία άλλη τεχνολογία δεν προσφέρει αυτήν την απανταχού παρουσία.

Ασφάλεια: Αν υποψιάζεστε ότι κάτι περίεργο συμβαίνει στον browser, η HTML5 και όλοι οι μοντέρνοι browsers που την τρέχουν έχουν εισάγει πολλά χαρακτηριστικά που ακολουθούν τα καινούργια πρότυπα ασφάλειας. Η εγγενής υποστήριξη σε προγράμματα περιήγησης για τα πολυμέσα και άλλες δυνατότητες μειώνουν την ανάγκη για επιπρόσθετα plugins μερικά από τα οποία έχουν τρωτά σημεία στον browser.

3.2 PHP

Η PHP είναι μια γλώσσα script από την πλευρά του διακομιστή , σχεδιασμένη ειδικά για το Web .Μέσα σε μια HTML σελίδα μπορείτε να ενσωματώσετε PHP κώδικα , που θα εκτελείται κάθε φορά που θα επισκέπτεστε τη σελίδα. Ο PHP κώδικας μεταφράζετε στο Web διακομιστή και δημιουργεί HTML ή άλλη έξοδο που θα δει ο επισκέπτης. Η PHP δημιουργήθηκε το 1994 και ήταν αρχικά η δουλειά ενός ατόμου , του Rasmus . Υιοθετήθηκε και από άλλα ταλαντούχα άτομα και έχει περάσει από τρεις εκδόσεις

Τον Ιανουάριο του 2001 ήταν σε χρήση σχεδόν σε πέντε τομείς παγκόσμια και αυτός ο αριθμός μεγαλώνει γρήγορα. Η PHP είναι ένα προϊόν ανοιχτού κώδικα. Έχουμε πρόσβαση στον κώδικα . Μπορούμε να τον χρησιμοποιήσου,ε , να τον αλλάξουμε και να τον αναδιανείνουμε , χωρίς χρέωση. PHP αρχικά σήμαινε Personal Home Page (προσωπική αρχική σελίδα), αλλά σύμφωνα με την σύμβαση GNU και τώρα σημαίνει PHP Hypertext (προεπεξεργαστής κειμένου PHP).

Ένα αρχείο με κώδικα PHP θα πρέπει να έχει την κατάλληλη επέκταση (π.χ. *.php, *.php4, *.phtml κ.ά.). Η ενσωμάτωση κώδικα σε ένα αρχείο επέκτασης .html δεν θα λειτουργήσει και θα εμφανίσει στον browser τον κώδικα χωρίς καμία επεξεργασία, εκτός αν έχει γίνει η κατάλληλη ρύθμιση στα MIME types του server. Επίσης ακόμη κι όταν ένα αρχείο έχει την επέκταση .php, θα πρέπει ο server να είναι ρυθμισμένος για να επεξεργάζεται κώδικα PHP. Ο διακομιστής Apache, που χρησιμοποιείται σήμερα ευρέως σε συστήματα με τα λειτουργικά συστήματα Linux και Microsoft Windows, υποστηρίζει εξ ορισμού επεξεργασία κώδικα PHP.

3.2.1 Πλεονεκτήματα της PHP

Κάποιοι από τους βασικούς ανταγωνιστές της PHP είναι ο Perl, Microsoft Active Server Pages (ASP) , Java Server Pages (JSP) και Allaire Cold Fusion .
Σε σύγκριση με αυτά τα προϊόντα, η PHP έχει πολλά πλεονεκτήματα όπως :

Υψηλή απόδοση: Η PHP είναι πολύ αποτελεσματική. Με ένα φθηνό διακομιστή μπορείτε να εξυπηρετήσετε εκατομμύρια επισκέψεων καθημερινά. Οι δοκιμές που δημοσιεύθηκαν από την Zend Technologies (http://www.zend.com), δείχνουν ότι η PHP ξεπερνά τους ανταγωνιστές της.

Διασυνδέσεις με πολλά διαφορετικά συστήματα βάσεων δεδομένων: Η PHP έχει εγγενείς συνδέσεις για πολλά συστήματα βάσεων δεδομένων. Εκτός από την MySQL μπορείτε να συνδεθείτε με τις βάσεις δεδομένων PostgreSQL , mSQL Oracle , dbm , filePro , Informix , InterBase , Sybase , κ.α. Χρησιμοποιώντας το Open Database Connectivity Standard (ODBC) μπορείτε να συνδεθείτε σε οποιαδήποτε βάση δεδομένων παρέχει ένα πρόγραμμα οδήγησης ODBC. Αυτό περιλαμβάνει και τα προϊόντα της Microsoft products , μεταξύ άλλων.

Ενσωματωμένες βιβλιοθήκες για πολλές συνηθισμένες Web διαδικασίες: Επειδή η PHP σχεδιάστηκε για να χρησιμοποιείται στο Web , έχει πολλές ενσωματωμένες βιβλιοθήκες , που εκτελούν πολλές χρήσιμες λειτουργίες σχετικές με το Web . Μπορείτε να δημιουργήσετε εικόνες GIF δυναμικά , να συνδεθείτε με άλλες υπηρεσίες δικτύων , να στείλετε ηλεκτρονικό ταχυδρομείο , να δουλέψετε με cookies και να δημιουργήσετε PDF έγγραφα : όλα αυτά με λίγες γραμμές κώδικα.

Ειναι Δωρεαν: Μπορείτε να κατεβάσετε την τελευταία έκδοση από το http://www.php.net , χωρίς χρέωση.

```php
$uri = $_SERVER['REQUEST_URI'];
$foo = $_GET['foo'];

header('Content-type: text/html');
echo 'Το URI που ζητήθηκε είναι: '.$uri;
echo 'Η τιμή της "foo" παράμετρου είναι: '.$foo;
```

Εικόνα 9

3.3 MySQL

Η MySQL έιναι ένα πολύ γρήγορο δυνατό, σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων. Μια βάση δεδομένων σας επιτρέπει να αποθηκεύετε, να αναζητάτε να ταξνομείτε και να ανακαλείται τα δεδομένα σας αποτελεσματικά. Ο MySQL διακοσμιτής ελεγχει την πρόσβαση στα δεδομένα σας για να διασφαλίσει οτι πολλοί χρήστες θα μπορούν να δουλεύουν ταυτόχρονα, για να υπάρχει γρήγορη πρόσβαση και για να διασφαλίσει ότι μόνο οι πιστοποιημένοι χρήστες θα μπορούν να έχουν πρόσβαση. Συνεπώς η MySQL έιναι ένας πολυνηματικός διακοσμιτής πολλαπλών χρηστών. Χρησιμοποιεί την SQL (Structured Query Language), την τυπική γλώωσα ερωτημάτων για βάσεις δεδομένων, παγκόσμια. Η MySQL έιναι διαθέσιμη απο το 1996, αλλά η ιστορία της ξεκινά απο το 1979. Είναι παγκοσμίως η πιο δημοφιλής βάση δεδομένων ανοικτού κωδικα και έχει κερδίσει το βραβείο Choice Award του Linux Journal Readers αρκετές φορές.

Ένα από τα μεγαλύτερα πλεονεκτήματα της MySQL είναι η δυνατότητα που δίνει για σύνδεση σε πολλές διαφορετικές βάσεις δεδομένων. Οι βάσεις δεδομένων που υποστηρίζονται περιλαμβάνουν τις: Adabas D, InterBase, PostgreSQL, dBase, FrontBase, SQLite, Empress, mSQL, Solid, FilePro (read-only), Direct MS-SQL, Sybase, Hyperwave, MySQL, Velocis, IBM, ODBC.

3.3.1 Πλεονεκτήματα MySQL

Μερικοί από τους κύριους ανταγωνιστές της MySQL είναι οι PostgreSQL , Microsoft SQL και Oracle. Η MySQL έχει πολλά πλεονεκτήματα , όπως χαμηλό κόστος , εύκολη διαμόρφωση και μάθηση και ο κώδικας προέλευσης είναι διαθέσιμος.

Απόδοση: Η MySQL είναι χωρίς αμφιβολία γρήγορη. Μπορείτε να δείτε την σελίδα δοκιμών http://web.mysql.com/benchmark.html . Πολλές από αυτές τις δοκιμές δείχνουν ότι η MySQL είναι αρκετά πιο γρήγορη από τον ανταγωνισμό.

Χαμηλό κόστος: Η MySQL είναι διαθέσιμη δωρεάν , με άδεια ανοικτού κώδικα (Open Source) ή με χαμηλό κόστος , αν πάρετε εμπορική άδεια, αν απαιτείται από την εφαρμογή σας.

Ευκολία Χρήσης: Οι περισσότερες μοντέρνες βάσεις δεδομένων χρησιμοποιούν SQL. Αν έχετε χρησιμοποιήσει ένα άλλο σύστημα διαχείρισης βάσεων δεδομένων δεν θα έχετε πρόβλημα να προσαρμοστείτε σε αυτό.

Μεταφερσιμότητα: Η MySQL μπορεί να χρησιμοποιηθεί σε πολλά διαφορετικά συστήματα Unix όπως επίσης και στα Microsoft Windows .

Κώδικας Προέλευσης: Όπως και με την PHP , μπορείτε να πάρετε και να τροποποιήσετε τον κώδικα προέλευσης της MySQL.
Νέα έκδοση :Η νέα έκδοση MySQL 5 έχει έρθει με νέες εντυπωσιακές λειτουργίες. Είναι πλέον ικανή να υποστηρίξει πολύ μεγάλα projects με υψηλή αξιοπιστία.

3.3.2 Τρόπος λειτουργίας MySQL Βάσης Δεδομένων

Ο τρόπος λειτουργίας της MySQL είναι ίδιος με αυτόν που ακολουθούν όλες οι Web βάσεις δεδομένων. Τα βήματα λειτουργίας των αρχιτεκτονικών των Web Βάσεων δεδομένων μπορούν να συνοψιστούν στα παρακάτω:

• Ο web browser ενός χρήστη κάνει μια HTTP αίτηση για μια συγκεκριμένη ιστοσελίδα. Για παράδειγμα, μπορεί να κάνει μια αναζήτηση για όλους τους χρήστες οι οποίοι είναι εγγεγραμμένοι σε ιστοσελίδα.
Έστω ότι η σελίδα των αποτελεσμάτων αναζήτησης ονομάζεται results.php

- Ο web διακομιστής λαμβάνει την αίτηση για τη σελίδα results.php, ανακαλεί το αρχείο και το περνά στην μηχανή PHP για επεξεργασία.
- Η μηχανή PHP αρχίζει την ανάλυση του script. Μέσα στο script, υπάρχει μια εντολή που συνδέει την βάση δεδομένων και εκτελεί ένα ερώτημα (την αναζήτηση των χρηστών). Η PHP ανοίγει μια σύνδεση με τον MySQL διακομιστή και στέλνει το κατάλληλο ερώτημα.
- Ο MySQL διακομιστής λαμβάνει το ερώτημα της βάσης δεδομένων και το επεξεργάζεται και στέλνει τα αποτελέσματα(μια λίστα χρηστών), ξανά στη μηχανή PHP.
- Η μηχανή PHP σταματά την εκτέλεση του script, που συνήθως περιλαμβάνει την μορφοποίηση των αποτελεσμάτων του ερωτήματος σε HTML. Επιστρέφει μετά την τελική HTML σελίδα στο web διακομιστή.
- Ο web διακομιστής περνά την HTML σελίδα ξανά στο browser, όπου ο χρήστης μπορεί να δει τη λίστα των σπουδαστών που ζήτησε.

3.4 Apache HTTP Server

Ο Apache http Server είναι ένας πολύ δημοφιλής διακομιστής διαδικτύου που διανέμεται ελεύθερα στο διαδίκτυο. Αναπτύχθηκε και συντηρείται από μια ομάδα εθελοντών που ήθελαν να υλοποιήσουν έναν εύρωστο κώδικα για διακομιστή δικτύου, που να μην είναι εμπορικός αλλά να υποστηρίζει πολλά χαρακτηριστικά.

Ο Apache όπως έχει αποδειχτεί είναι ο πιο γρήγορος, σταθερός, ασφαλής και υποστηρίζει τα περισσότερα χαρακτηριστικά από οποιονδήποτε άλλο διακομιστή δικτύου. Ο Apache είναι εγκατεστημένος στο 80% των διακομιστών παγκοσμίως (πάνω από 6 εκατομμύρια διακομιστές). Πάνω του είναι εγκατεστημένα εκατομμύρια sites που δέχονται εκατομμύρια hits καθημερινά χωρίς να παρουσιάζεται κανένα απολύτως πρόβλημα.

Σήμερα ο Apache θεωρείται από τους πιο σταθερούς διακομιστές δικτύου που κυκλοφορούν και θα πρέπει να τονίσουμε ότι αρκετοί εμπορικοί διακομιστές διαδικτύου, όπως ο HTTP Server της IBM, χρησιμοποιούν τον πυρήνα του Apache.

3.4.1 Χαρακτηριστικά του Apache 2

- Unix treading: Υποστήριξη συστημάτων Unix με νήματα POSIX, όπου ο Apache μπορεί να “τρέχει” πολλές διεργασίες ταυτόχρονα.
  Υποστήριξη πολλαπλών πρωτοκόλλων: Υποστηρίζει πιο γρήγορα και πιο σταθερά λειτουργικά όπως BeOS, OS/2 και Windows.
- Φίλτρα: Υποστήριξη φίλτρων που διανέμονται από και προς τους διακομιστές.
- Λάθη: Τα μηνύματα λαθών μπορούν να εμφανίζονται σε διάφορες γλώσσες.
  Απλοποιημένη παραμετροποίηση: Έχουν απλοποιηθεί κάποια directives που ως τώρα ήταν κάπως μπερδεμένα.
- Υποστήριξη unicode: Ο Apache 2 σε Windows NT χρησιμοποιεί μόνο utf−8 κωδικοποίηση.
  Κανονικές εκφράσεις: Υποστήριξη της βιβλιοθήκης PCRE δηλαδή όλες οι κανονικές εκφράσεις που υποστηρίζει η Perl 5.

3.5 CSS

CSS σημαίνει Cascading Style Sheets και είναι στυλ που μπορούμε να ορίσουμε για τις HTML σελίδες. Γράφοντας τις σελίδες μας μόνο με HTML κώδικα, μπορούμε να ορίσουμε το χρώμα και το μέγεθος του κειμένου αλλά και άλλων στοιχείων της σελίδας (όπως πίνακες, links, λίστες κτλ). Για να αλλάξουμε το χρώμα κάποιου κειμένου ή το χρώμα ενός πίνακα, θα πρέπει να βρούμε το χρώμα αυτό μέσα στον κώδικα και να το αλλάξουμε. Η διαδικασία αυτή μπορεί να φαντάζει εύκολη όταν έχουμε να διαχειριστούμε μια μόνο σελίδα, αλλά ένα site αποτελείται από δεκάδες σελίδες τις οποίες χρειάζεται να διαχειριζόμαστε εύκολα και γρήγορα. Φανταστείτε, για παράδειγμα, πόσο χρονοβόρο θα είναι αν θελήσουμε κάποια στιγμή να αλλάξουμε τα χρώματα στο κύριο μενού του site μας, το οποίο επαναλαμβάνεται σε όλες τις σελίδες. Σε μια τέτοια περίπτωση θα χρειαζόταν να ανοίγουμε κάθε σελίδα του site και να αλλάζουμε τα χρώματα του φόντου και των links του μενού, διαδικασία που εκτός από χρονοβόρα είναι και κουραστική.

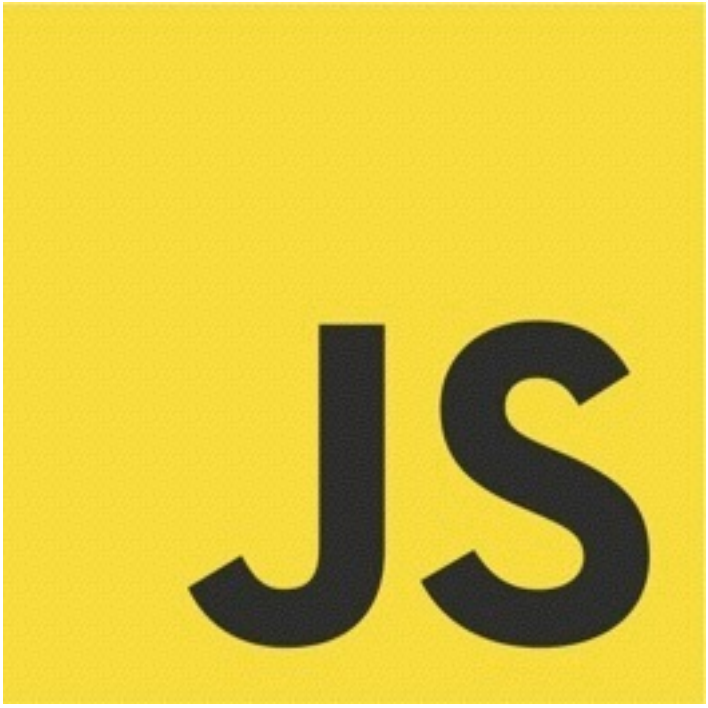
Με την χρήση CSS μπορούμε να ορίζουμε χρώματα και μεγέθη οργανωμένα σε στυλ και έπειτα να εφαρμόζουμε τα στυλ αυτά στα στοιχεία των σελίδων του site μας. Με αυτόν τον τρόπο, κάθε φορά που αλλάζουμε το χρώμα ενός στυλ, αλλάζει το χρώμα όλων των στοιχείων που έχουν αναφορά στο στυλ αυτό. Έτσι αν έχουμε ορίσει ένα στυλ για το κύριο μενού του site, τότε θα χρειάζεται να αλλάξουμε το χρώμα του στυλ αυτού και αυτόματα θα εφαρμοστεί σε όλες τις σελίδες.

Εκτός από την ευκολία στην διαχείριση ενός site, ένα άλλο σημαντικό πλεονέκτημα της χρήσης CSS στις σελίδες είναι ο "καθαρότερος" κώδικας, χωρίς πολλές ιδιότητες στις ετικέτες οι οποίες τον κάνουν δυσανάγνωστο. Επιπλέον κάνει γρηγορότερη την πλοήγηση καθώς το αρχείο, μέσα στο οποίο ορίζονται τα στυλ, "διαβάζεται" από τον browser μόνο μια φορά και έπειτα αποθηκεύεται στην cache memory, μειώνοντας έτσι το μέγεθος της πληροφορίας που γίνεται download από τους browsers.

Η CSS3 είναι η εξέλιξη της CSS και είναι απόλυτα συμβατή με όλες τις προγενέστερες εκδόσεις της, συνεπώς δεν χρειάζεται να αλλάξετε κώδικα που έχετε ετοιμάσει με προηγούμενη έκδοση. Η CSS3 είναι χωρισμένη σε modules. Τα παλιά χαρακτηριστικά της έχουν διασπαστεί σε μικρότερα κομμάτια και έχουν προστεθεί και νέα. Κάποια από τα σημαντικότερα είναι:

- Selectors
- Box Model
- Backgrounds and Borders
    - Text Effects
- 2D/3D Transformations
- Animations
- Multiple Column Layout
    - User Interface

3.6 Javascript



Η JavaScript (JS) είναι διερμηνευμένη γλώσσα προγραμματισμού για ηλεκτρονικούς υπολογιστές.Αρχικά αποτέλεσε μέρος της υλοποίησης των φυλλομετρητών Ιστού, ώστε τα σενάρια από την πλευρά του πελάτη (client-side scripts) να μπορούν να επικοινωνούν με τον χρήστη, να ανταλλάσσουν δεδομένα ασύγχρονα και να αλλάζουν δυναμικά το περιεχόμενο του εγγράφου που εμφανίζεται.

Η JavaScript είναι μια γλώσσα σεναρίων που βασίζεται στα πρωτότυπα (prototype-based), είναι δυναμική, με ασθενείς τύπους και έχει συναρτήσεις ως αντικείμενα πρώτης τάξης. Η σύνταξή της είναι επηρεασμένη από τη C. Η JavaScript αντιγράφει πολλά ονόματα και συμβάσεις ονοματοδοσίας από τη Java, αλλά γενικά οι δύο αυτές γλώσσες δε σχετίζονται και έχουν πολύ διαφορετική σημασιολογία. Οι βασικές αρχές σχεδιασμού της JavaScript προέρχονται από τις γλώσσες προγραμματισμού Self και Scheme.[22] Είναι γλώσσα βασισμένη σε διαφορετικά προγραμματιστικά παραδείγματα (multi-paradigm), υποστηρίζοντας αντικειμενοστρεφές, προστακτικό και συναρτησιακό[23] στυλ προγραμματισμού.

Η JavaScript χρησιμοποιείται και σε εφαρμογές εκτός ιστοσελίδων — τέτοια παραδείγματα είναι τα έγγραφα PDF, οι εξειδικευμένοι φυλλομετρητές (site-specific browsers) και οι μικρές εφαρμογές της επιφάνειας εργασίας (desktop widgets). Οι νεότερες εικονικές μηχανές και πλαίσια ανάπτυξης για JavaScript (όπως το Node.js) έχουν επίσης κάνει τη JavaScript πιο δημοφιλή για την ανάπτυξη εφαρμογών Ιστού στην πλευρά του διακομιστή (server-side).

[22] ECMAScript Language Overview» (PDF). 2007-10-23, σ. 4

[23] Douglas Crockford on Functional JavaScript

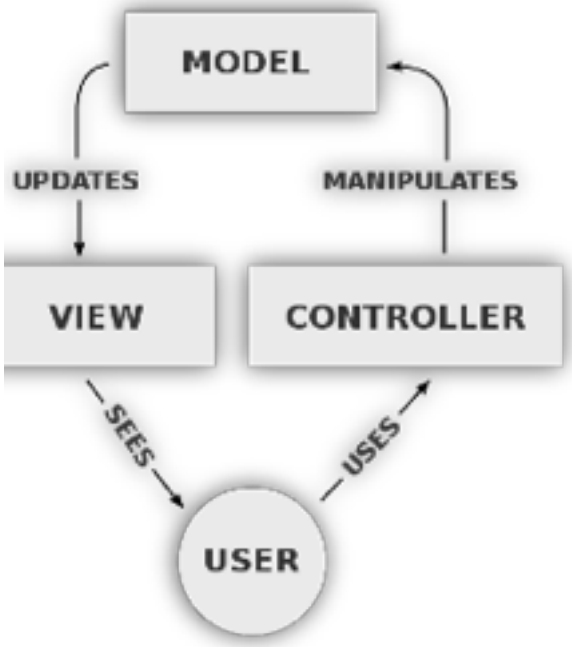
Το πρότυπο της γλώσσας κατά τον οργανισμό τυποποίησης ECMA ονομάζεται ECMAscript

ΚΕΦΑΛΑΙΟ 4 Αρχιτεκτονική- Θεωρία Εφαρμογής

Σε αυτή την ενότητα θα γίνει μια ειδική αναφορά πάνω στην Αρχιτεκτονική στην οποία έχει στηριχθεί ο προγραμματισμός της εφαρμογής και της ίδιας πλατφόρμας προγραμματισμου.

4.1 MVC (Model View Controller)

Το MVC (Model-View-Controller) είναι ένα σχεδιαστικό πρότυπο με συγκεκριμένη αρχιτεκτονική η οποία βασίζεται στην ιδέα της αντικειμενοστρέφειας (Object Oriented). Η κεντρική ιδέα του MVC έγκειται στο γεγονός ότι χρησιμοποιεί κλάσεις για να οργανώσει το Business Logic (που αποθηκεύονται τα δεδομένα , ποιός τα χειρίζεται, και πώς αυτά τίθενται σε επεξεργασία) πού περιέχεται στο Model, το Presentation Logic (πώς τα δεδομένα από το Model θα αποδοθούν) στα Views και έχει μία συνολική ροή της εφαρμογής μέσα από τους Controllers. Στο MVC γενικά, το model αναπαριστά τα δεδομένα-πληροφορίες της εφαρμογής , το view αποτελείται από ένα σύνολο HTML, CSS , εικόνες κλπ και έχει να κάνει με το πώς εμφανίζονται οι πληροφορίες αυτές στην διεπαφή του χρήστη και τέλος το controller χειρίζεται και ελέγχει την επικοινωνία του model με το view. Παρακάτω παρουσιάζεται ένα διάγραμμα του MVC και πως επικοινωνούν τα συστατικά του.



Εικόνα 10

4.2 Πλεονεκτήματα Σχεδιαστικού Προτύπου MVC

Ένα σχεδιαστικό πρότυπο είναι μία συλλογή από αντικείμενα, και σχέσεις μεταξύ αυτών, το οποίο έχει αναπτυχθεί και δοκιμαστεί για να επιλύει αποτελεσματικά μία ιδιαίτερη κατηγορία προβλημάτων. Εάν ένα ενδεχόμενο πρόβλημα που χρειάζεται λύση ταιριάξει με ένα που έχει ήδη λυθεί από ένα σχεδιαστικό πρότυπο, κατόπιν παρουσιάζεται ένα σημαντικό πλεονέκτημα στην υλοποίηση της ζητούμενης λύσης με την υιοθέτηση του σχεδίου αυτού.

Γενικότερα, το βασικότερο πλεονέκτημα του προτύπου MVC είναι ότι η αρχιτεκτονική του συστήματος χωρίζεται σε επιμέρους τμήματα τα οποία είναι ανεξάρτητα μεταξύ τους. Το γεγονός αυτό δίνει την δυνατότητα στους προγραμματιστές να μπορούν να εστιάζουν κάθε φορά σε ένα ξεχωριστό τμήμα της εφαρμογής, επηρεάζοντας σημαντικά την μείωση του χρόνου για την διόρθωση των λαθών (debugging).

Ένα ακόμα χαρακτηριστικό γνώρισμα της αρχιτεκτονικής MVC είναι η ευελιξία που προσδίδει στο σύστημα που την χρησιμοποιεί. Μιλώντας για ευελιξία αναφερόμαστε στις δυναμικές αλλαγές που μπορούν να συμβούν και στην ευκολία πραγματοποίησης τους. Με άλλα λόγια, η αρχιτεκτονική αυτή επιτρέπει τυχόν προσθέσεις, τροποποιήσεις ή απαλοιφής κάποιου view με εύκολο τρόπο, ή ακόμα και αλλαγή σε κάποια απόκριση του controller. Ακόμα και στην περίπτωση όπου ο σχεδιαστής επιθυμεί να αλλάξει τύπο βάσης δεδομένων π.χ από MySQL σε Oracle το μόνο που χρειάζεται να τροποποιηθεί είναι το model και τίποτε άλλο.

Παράλληλα, λόγω της ιδιομορφίας που παρουσιάζει το πρότυπο MVC, δίνεται η ευχέρεια στους εκάστοτε προγραμματιστές να μπορούν να συνδυάζουν ποικίλες τεχνολογίες για την αναπαράσταση των τελικών δεδομένων. Αυτό συνεπάγεται ότι για την εμφάνιση μίας πληροφορίας (model) μπορούν χρησιμοποιηθούν πολλαπλές παρουσιάσεις (views) , χωρίς να μεσολαβεί κάποια απαραίτητη επέμβαση από τον χρήστη. Αυτό αυτόματα σημαίνει ότι δεν έχει σημασία πώς ο χρήστης επιθυμεί να του εμφανίζεται η πληροφορία π.χ Flash , WAP , διότι και οι δύο τεχνολογίες αναφέρονται στο ίδιο model

Τέλος, η αρχιτεκτονική του MVC εξυπηρετεί επίσης και στην επαναχρησιμοποίηση του κώδικα της εφαρμογής όποτε αυτό χρειαστεί (σε περίπτωση όπου ένα view χρησιμοποιεί διαφορετικά model για να εμφανίσει πληροφορία). Συγχρόνως παρέχει ελευθερία στους σχεδιαστές του συστήματος, δίνοντάς τους την δυνατότητα να επεξεργάζονται τα συστατικά του MVC ταυτόχρονα χωρίς να επηρεάζεται ο υπόλοιπος κώδικας.

4.3 Στοιχεία του MVC

4.3.1 Model

Το κομμάτι του model αποτελεί τον πυρήνα της εφαρμογής και είναι υπεύθυνο για την υλοποίηση του Business Logic. Το Business Logic αναφέρεται σε οποιαδήποτε δραστηριότητα σχετιζόμενη με το πώς

αποθηκεύονται τα δεδομένα σε μία εφαρμογή. Βασική του αρμοδιότητα είναι η αποθήκευση των δεδομένων της εφαρμογής σε συγκεκριμένο χώρο π.χ βάση δεδομένων όπως επίσης και η ενημέρωση των views όταν λάβει χώρα κάποια αλλαγή σε αυτό.

Το όνομα model προέρχεται από το γεγονός ότι τα αντικείμενα εξομοιώνουν πράγματα από τον πραγματικό κόσμο, όπως κανόνες και δεδομένα. Τα models είναι το μόνα συστατικά του συστήματός που αλληλεπιδρούν με τα δεδομένα (ιδιαίτερη ονοματολογία για καθετί που αποθηκεύει και ανακτά τα δεδομένα μεταξύ των αιτημάτων). Επιπλέον, μπορούν να υλοποιήσουν οποιανδήποτε από τις τεχνολογίες

που κάνουν την γλώσσα PHP μία σπουδαία γλώσσα για την ανάπτυξη εφαρμογών Ιστού όπως: κλήσεις σε βάσεις δεδομένων, χειρισμός αρχείων, διαχείριση
συνόδου, υπηρεσίες Ιστού κ.α.

Ένα καλά σχεδιασμένο model θα μπορούσε να χρησιμοποιηθεί για άλλες εφαρμογές σε έναν εξυπηρετητή (web server). Όταν οι προγραμματιστές διαπιστώσουν την δύναμη της τοποθέτησης του Business Logic σε αυτές τις κλάσεις, αυτό που γίνεται σαφές είναι ότι η παραγωγή καλοσχεδιασμένων models είναι πολύ χρήσιμη πρακτική, ακόμα κι αν δεν χρησιμοποιείται σε μεγάλο βαθμό η MVC αρχιτεκτονική.

Για τα model που αλληλεπιδρούν με έναν πίνακα βάσεων δεδομένων, ένας ανεξάρτητος πίνακας πρέπει να συνδεθεί ιδανικά με μόνο κλάση model. Αυτό αποτελεί και ένα από τα σημαντικά οφέλη του MVC: διατήρηση των δεδομένων σε κάποιο είδος αποθηκευτικού μέσου.

Οι προγραμματιστές παραμένοντας προσκολλημένοι στην λογική του MVC, θα χρειαστεί μόνο να αναθεωρήσουν τα models του σύστήματος εάν αλλάξει ένας πίνακας βάσεων δεδομένων. Με αυτόν τον τρόπο, θα υπήρχε μία κλάση μόνο που θα ήταν αναγκαίο να ανανεωθεί.

Παραδείγματα χαρακτηριστικών models στις εφαρμογές Ιστού μπορούν να είναι: ο χρήστης, η μηχανή αναζήτησης, ένα καλάθι αγορών, ένας κατάλογος με στοιχεία, παραγγελίες, θέματα ή άρθρα. Σύμφωνα με διάφορες μελέτες διαπιστώθηκε ότι για τις εφαρμογές Ιστού, τα models τείνουν να σχετίζονται με έναν μοναδικό πίνακα, ή με μία μικρή ομάδα στενά συνδεμένων πινάκων. Το όνομα της κλάσης του model είναι πιθανό να είναι παρόμοιο με το όνομα του πίνακα, προϋποθέτοντας όμως ότι βρίσκεται στα πλαίσια των συμβάσεων ονομάτων που ισχύουν.

4.3.2. View

Το view είναι το συστατικό της MVC αρχιτεκτονικής το οποίο αποδίδει τα δεδομένα που είναι αποθηκευμένα στο model με τον κατάλληλο τρόπο για την αλληλεπίδραση τους με τον χρήστη . Για μία εφαρμογή Ιστού , το view θα μπορούσε να είναι μία σελίδα σε HTML, η οποία θα παρουσιάζει την πληροφορία που υπάρχει στο model. Η MVC εφαρμογή μπορεί να εξαγάγει: XML, WML, απλό κείμενο, εικόνες, ηλεκτρονικό ταχυδρομείο ή κάποιο άλλο περιεχόμενο. Επιπρόσθετα το view είναι υπεύθυνο να ανανεώνει την έξοδο του (interface) αν συμβεί κάποια αλλαγή στο model καθώς και να προωθεί την είσοδο από τον χρήστη (request) στον controller με σκοπό να ενεργοποιηθούν οι απαραίτητες διεργασίες για να ικανοποιηθεί το αίτημα του χρήστη.

4.3.3 Controller

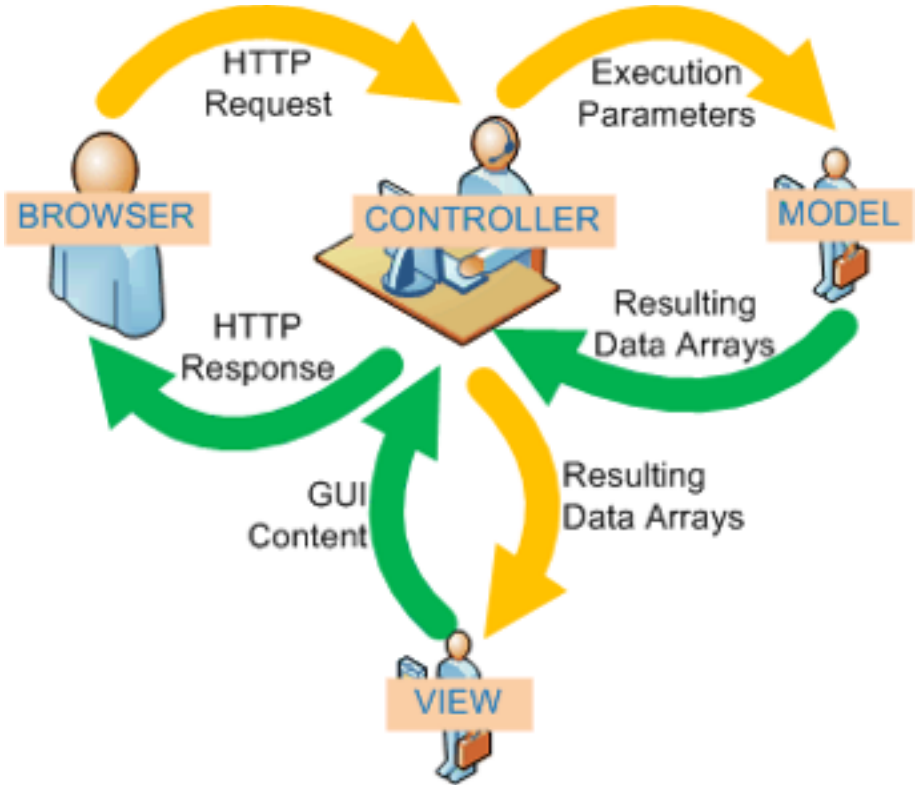
Ο controller είναι το συστατικό εκείνο που χειρίζεται το κάθε αίτημα του χρήστη π.χ HTTP request. Όταν λάβει μία τέτοιου είδους αίτηση ο controller θα προσδιορίζει τί περιλαμβάνει αυτή η αίτηση από τον συγκεκριμένο χρήστη και θα αποκριθεί κατάλληλα . Ειδικότερα, θα ενεργοποιήσει το model να διαχειριστεί τα δεδομένα και αυτό με την σειρά του να τα προωθήσει στο view.

Από τα παραπάνω καταλήγουμε στο ότι ο controller πρέπει να καθορίζει ποιό view πρόκειται να εμφανίσει, ή τι άλλη δράση σχετική με την εφαρμογή πρέπει να πραγματοποιηθεί. Βασικός κανόνας είναι ότι ποτέ δεν πρέπει να έχουμε κάποια HTML σε ένα model, όπως επίσης οποιεσδήποτε λειτουργίες πρόσβασης βάσεων δεδομένων στα views. Εφαρμόζοντας κάποια από αυτά θα ήταν μία παραβίαση της αρχής διαχωρισμού του περιεχομένου από την μορφοποίηση, καταλήγοντας στην καταστροφή του MVC μοντέλου και περιορίζοντας την ευελιξία της εφαρμογής.

4.4 Επικοινωνία στοιχείων MVC

Παρακάτω παρατίθεται μία αναλυτικότερη σχέση μεταξύ των τριων συστατικών του MVC και πως αυτά επικοινωνούν όταν ο χρήστης ζητήσει κάποια αίτηση.

1) Αρχικά ο χρήστης στέλνει το αίτημα του μέσω ενός Web Browser π.χ Http Request .
2) Έπειτα το view μέσω του interface στέλνει την είσοδο στον controller
3) Ο controller λαμβάνει την είσοδο από το view.
4) Ο controller τροποποιεί το model στέλνοντας τις απαραίτητες παραμέτρους σύμφωνα με την είσοδο του χρήστη (user input).
5) Το model αλλάζει , βασισμένο στην ανανέωση που προκάλεσε ο controller σε αυτό. Το model



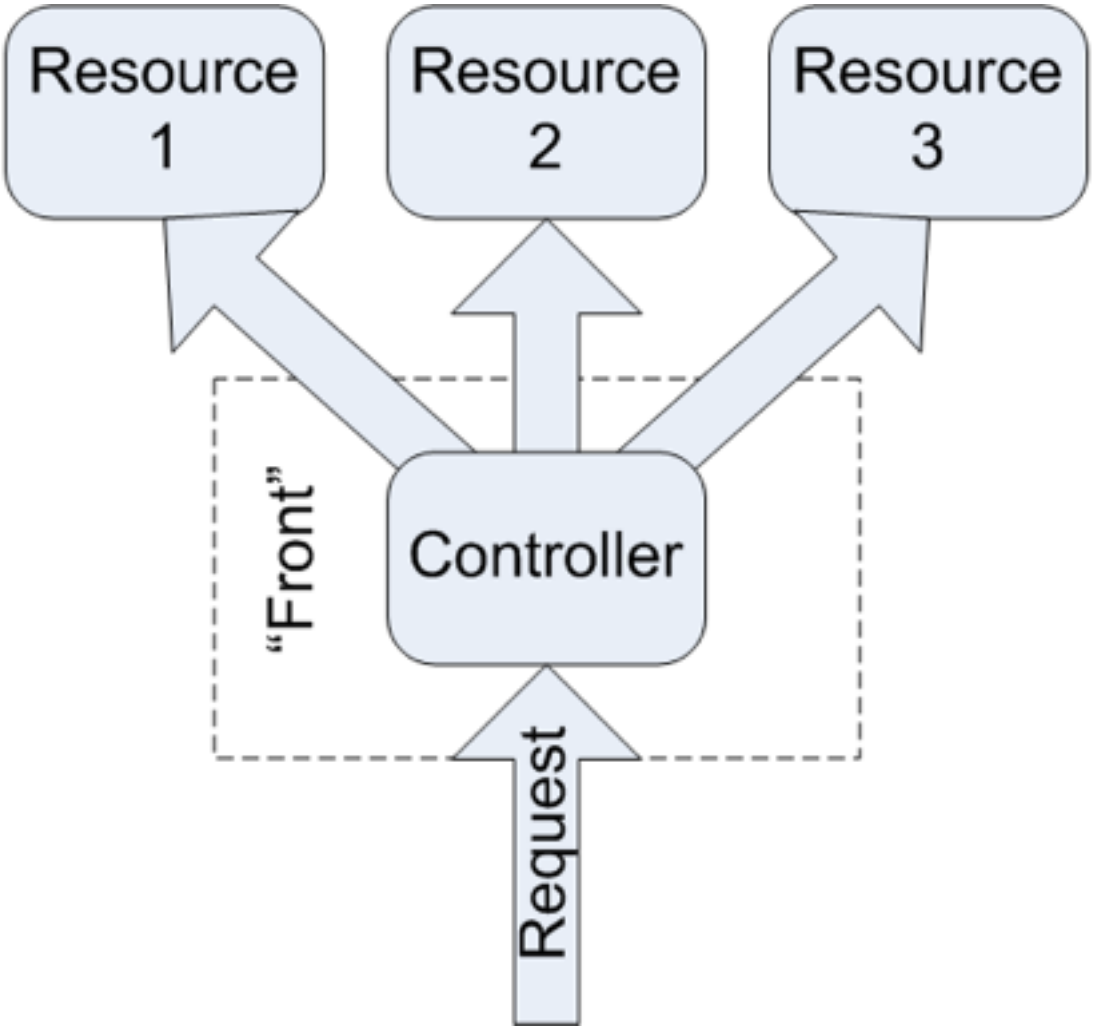
ενημερώνει το view για την αλλαγή αυτή που υπέστη.
6) Το view ανανεώνει το user interface του χρήστη και συνεπώς απαντά στο αίτημα του.

Εικόνα 10

Σε μερικές περιπτώσεις ο controller τροποποιεί το view κατευθείαν και δεν ανανεώνει το model καθόλου.

4.5 Front Controller

Ο Front Controller pattern είναι ένα σχεδιαστικό πρότυπο του επιπέδου παρουσίασης μιας εφαρμογής. Χρησιμοποιείται για να απλοποιήσει την αναπαράστασης των δεδομένων εφαρμογής και να διαχειρίζεται την επικοινωνία μεταξύ του client (πελάτη) και των διαφόρων συνιστωσών της εφαρμογής (server). Όταν μιλάμε για τα σχεδιαστικά πρότυπα που χειρίζονται την αλληλεπίδραση με το χρήστη, μιλάμε για Action Control Pattern.



Εικόνα 12

Ο Front Controller διαχειρίζεται την επικοινωνία μεταξύ του χρήστη και τις διάφορες συνιστώσες της εφαρμογής, παρέχοντας ένα περιβάλλον που χειρίζεται όλα τα αιτήματα των χρηστών. Αυτό σημαίνει, ότι όλα τα requests που αποστέλλονται στην web εφαρμογή επεξεργάζονται απο τον FrontController, ο οποίος στη συνέχεια ανακατευθύνει κάθε request στον αντίστοιχο resource που αντιστοιχεί σε κάποιον ελεγκτή δράσης (action controller). Επίσης ένας Front controller θα μπορούσε να υλοποιήσει μηχανισμούς για κάθε request προτού το request αυτο δρομολογηθεί στο στοχευμένο ελεγκτή δράσης απο τον Front Controller. Με αυτό τον τρόπο ο Front Controller μπορεί να ελέγχει και να διακόπτει την πρόσβαση σε μη πιστοποιημένους Χρήστες καθώς και να κρατάει ιστορικό για όλα τα requests με σκοπό την καλύτερη κατανόηση των αιτηματων των χρηστών σε μια εφαρμογή.

Συνοψίζοντας ο Front Controller αποτελεί ένα κεντρικό σημείο εισόδου και διαχείρισης για όλα τα request του client, διεκπεραιώνοντας εργασίες οπως φιλτράρισμα εισόδου και και προσωρινής αποθήκευσης δεδομένων και διαχείρισης τους.

ΚΕΦΑΛΑΙΟ 5 Υλοποίηση Αρχιτεκτονικής MVC Πλατφόρμας

Ένα απο τα χαρακτηριστικά των MVC είναι η Δομή των διευθύνσεων URL με την οποία μια εφαρμογή ή ένα site λειτουργεί. Η δομή που πρέπει να ακολουθήσει η εφαρμογή μας είναι http://domain/controller/action/id, οπου το action και id είναι προαιρετικά. Για να το καταφέρουμε αυτό θα πρεπει να χρησιμοποιήσουμε ένα εξωτερικό αρχείο server που χρησιμοποιεί ο Apache το .htaccess URL re-writing, το URL re-writing έρχεται με ενα εσωτερικό module του apache το 'mod_rewrite'.

Για να αξιοποιήσουμε τη λειτουργία του mod_rewrite, θα χρειαστεί να δημιουργήσουμε ένα αρχείο με το όνομα .htaccess. Αυτό το αρχείο θα τοποθετηθει στον κατάλογο root της εφαρμογής μας. Στην περίπτωσή μας, .htaccess θα μοιάζει με την παρακάτω εικόνα.

```
Options +FollowSymLinks
RewriteEngine on
RewriteRule ^([a-zA-Z]*)/?([a-zA-Z]*)?/?([a-zA-Z0-9]*)?/?$ in-
dex.php?controller=$1&action=$2&id=$3 [NC,L]
```

Εικόνα 13

Οι δύο πρώτες γραμμές προετοιμάζουν τον Apache για τον rewrite κανόνα, τον οποίο τον ορίζουμε κάνοντας χρήση της RewriteRule εντολής. Το πρώτο μερος της εντολής αυτής είναι regular expression και στο δεύτερο μέλος ακολουθεί μια γενική διευθυνση URL με κάποιες μεταβλητές. Αυτο που μεταφράζεται απο την εντολή, ειναι οτι οποιαδήποτε url ο χρήστης εισάγει που ταιριάζει με το regular expression, αυτομάτως μεταφράζεται σε αληθινή μορφή επιπέδου web server (index.php?controller=home).

Ένα βασικό συστατικό αυτού του κανόνα είναι η αναφορά του στο index.php. Το index.php είναι μια σελίδα προορισμού (landing page) για όλες τις αιτήσεις (requests) στο πλαίσιο MVC, που σε συνδυασμό με το υποπρογραμμα loader.php που θα δουμε στη συνέχεια αποτελουν το MVC συστατικό τον Front Controller.

```
/* index.php
------------ */
<?php
//load the required classes
require("classes/basecontroller.php");
require("classes/basemodel.php");
require("classes/view.php");
require("classes/viewmodel.php");
require("classes/loader.php");
require('libs/validator.php');
require('libs/user.php');

$loader = new Loader(); //create the loader object
$controller = $loader->createController(); //creates the requested controller object
based on the 'controller' URL value
$controller->executeAction(); //execute the requested controller's requested method
based on the 'action' URL value. Controller methods output a View.
?>
```

Εικόνα 14

Συνοψίζοντας, το index.php "απαιτεί" και φορτωνει με την βοηθεια του υποπρογράμματος "loader" όλα ταυποπρογράμματα που χρειάζονται και στη συνέχεια σε τρεις γραμμες γίνεται η εγκατάσταση του ελεγκτή (controller) και ή εκτέλεση της ενέργειας (action).

5.1 Η κλάση Loader

Όταν ένα στιγμιότυπο της κλάσης loader έχει δημιουργηθεί, περνάει ως όρισμα τις url παραμέτρους, οι οποίες έχουν κρατηθεί απο την PHP στον υπερπίνακα $_GET. Ο loader constructor (ο οποίος καλείται αυτόματα κατα τη δημιουργία στιγμιοτύπου) αποθηκευει την $_GET σε μια παράμετρο και στη συνέχεια εξάγει τα ζητουμενα URL δεδομένα (action, controller) με τη βοήθεια συνθηκών If-else σε private μεταβλητες της κλάσης.
Έχοντας αυτές τις μεταβλητές γίνεται ελεγχος για το αν υπάρχουν τα ζητούμενα resources στο σύστημα,

```php
<?php
class Loader {

    private $controllerName;
    private $controllerClass;
    private $action;
    private $urlValues;

    //store the URL request values on object creation
    public function __construct() {
        $this->urlValues = $_GET;
        //die(print_r($this->urlValues));

        if ($this->urlValues['controller'] == "") {
            $this->controllerName = "home";
            $this->controllerClass = "HomeController";
        } else {
            $this->controllerName = strtolower($this->urlValues['controller']);
            $this->controllerClass = ucfirst(strtolower($this->urlValues['controller'])) . "Controller";
        }

        if ($this->urlValues['action'] == "") {
            $this->action = "index";
        } else {
            $this->action = $this->urlValues['action'];
        }
    }

    //Factory method which establishes the requested controller as an object
    public function createController() {
        //check our requested controller's class file exists and require it if so
        if (file_exists("controllers/" . $this->controllerName . ".php")) {
            require("controllers/" . $this->controllerName . ".php");
        } else {
            require("controllers/error.php");
            return new ErrorController("baduri",$this->urlValues);
        }

        //does the class exist?
        if (class_exists($this->controllerClass)) {
            $parents = class_parents($this->controllerClass);

            //does the class inherit from the BaseController class?
            if (in_array("BaseController",$parents)) {
                //does the requested class contain the requested action as a method?
                if (method_exists($this->controllerClass,$this->action))
                {
                    return new $this->controllerClass($this->action,$this->urlValues);
                } else {
                    //bad action/method error
                    require("controllers/error.php");
                    return new ErrorController("baduri",$this->urlValues);
                }
            } else {
                //bad controller error
                require("controllers/error.php");
                return new ErrorController("baduri",$this->urlValues);
            }
        } else {
            //bad controller error
            require("controllers/error.php");
            return new ErrorController("baduri",$this->urlValues);
        }
    }

    public function get() {
        echo 'asdasd';
    }
}

?>
```

Εικόνα 14

έλεγχος όπως για το αν υπάρχει ο ζητούμενος ελεγκτής, αν ο ελεγτής αυτός υιοθετεί μεθόδους και παραμέτρους που προαιπαιτουνται για τη βασική λειτουργία της πλατφόρμας απο την BaseController κλαση για την οποία θα αναφερθούμε στην επόμενη ενότητα. Μετα απο τους έλεγχους και αναλογα το αποτέλεσμα "φορτωνει" το αρχειο php για τον ζητούμενο ελεγκτή και τη μέθοδο του.

5.2 Ελεγκτές (controllers)

Στη κλάση Loader έιχαμε σημειώσει οτι γίνεται έλεγχος για το αν η κλάση του ζητούμενου ελεγκτή υιοθετεί μεθόδους και παραμέτρους απο την BaseController class. Η BaseController class έιναι η parent class όλων των ελεγκτών. Ο σκοπός του 'ελεγχου αυτού έιναι να σιγουρέψουμε οτι για κάθε αντικείμενο controller που δημιουργούμε , να εξασφαλίζουμε οτι έχει υιοθετήσει τους παραμέτρους και τις μεθόδους της BaseController class.

```php
<?php
abstract class BaseController {

    protected $urlValues;
    protected $action;
    protected $model;
    protected $view;

    public function __construct($action, $urlValues) {
        $this->action = $action;
        $this->urlValues = $urlValues;

        //establish the view object
        $this->view = new View(get_class($this), $action);
    }

    //executes the requested method
    public function executeAction() {
        return $this->{$this->action}();
    }
```

Εικόνα 14

Εκτός απο την executeAction μεθοδο η οποία εκτελεί την μέθοδο/ενέργεια του request του στιγμιοτύπου controller που δημιούργησε ο loader, η κλαση αυτή στον constructor της δημιουργεί το αντικείμενο View το οποίο παρουσιάζει την πληροφορία που παίρνει ο controller απο το model (θα αναφερθουμε στη συνέχεια για αυτα τα δύο).

Η ουσία είναι οτι μαζί με τις μεθόδους του BaseController δημιουργουμε και αντικείμενο View μέσα απο αυτο έτσι ώστε οι ελεγκτές που θα υιοθετούν αυτη τον BaseController θα μπορουν μέσω αυτού να καλούν και τις μεθοδους του View αντικειμένου χωρις να δημιουργούνε ρεπλικες για κάθε controller που δημιουργήται στο μελλον.

Οι κλάσεις που γράφουμε για του πραγματικούς ελεγκτές θα πρέπει να κάνουν extend τον BaseController όπως ο παρακάτω ελεγκτής Home που κάνει ακριβώς αυτό (ο οποιος βρίσκεται στο controllers/home.php κατάλογο).

```php
<?php
class HomeController extends BaseController
{
    //add to the parent constructor
    public function __construct($action, $urlValues) {
        parent::__construct($action, $urlValues);

        //create the model object
        require("models/home.php");
        $this->model = new HomeModel();
    }

    //default method
    protected function index()
    {
        $this->view->output($this->model->index());
    }
}

?>
```

Εικόνα 16

Αυτή έιναι μια πολύ απλή κλάση ελεγκτή, οπως παρατηρείτε επεκτείνει τον BaseController και οριζει την μέθοδο “index”, με την οποία καλούμε την μέθοδο output() του View Αντικειμένου (που έχει δημιουργηθεί στον BaseController και οι μεθόδοι είναι διαθέσιμοι σε όποιον ελεγκτη τον κάνει extend) , με όρισμα τα αποτελέσματα της μεθόδου index του HomeModel τα οποία ειναι η πληροφορία που θέλουμε να επιστρεψουμε στο request του Client.

Οι μέθοδοι του controller είναι υπέυθυνοι στο να δημιουργούν στιγμιοτυπα του σχετικού model και να επιστρέφουν τα επιστρεφόμενα δεδομένα αυτού με το συσχετισμένο template View.

5.3 Οργάνωση και αποτύπωση πληροφορίας (Models)

Ομοίως με την BaseController class, οι μέθοδοι και οι παράμετροι της BaseModel class (σχημα 4.3.1) υιοθετουνται απο τα αντίστοιχα στιγμιότυπα model που δημιουργουνται μεσα απο τον αντίστοιχο ελεγκτή όπως και τα δεδομένα επιστρέφουν μεσα απο το αντίστοιχο στιγμιότυπο view.

```php
<?php

class BaseModel{

    protected $viewModel;
    protected $database;
    protected $validator;
    protected $url;
    protected $offset = 2; // we use max 3 segment type url


    //create the base and utility objects available to all models on model creation
    public function __construct()
    {
        $this->viewModel = new ViewModel();
        $this->commonViewData();
        $this->_db = new PDO("mysql:host=localhost;dbname=ebook", "ebook", "123");
        $this->validator = new Validation();
        $this->userservice = new UserService($this->_db, $email='', $passwd='');

    }

```

Εικόνα 17

Στην εικόνα 16 στον constructor του HomeController δημιουργήται το αντίστοιχο στιγμιότυπο HomeModel (εικόνα 19) έτσι ώστε μέσα απο τον ελεγκτή να μπορέσουμε να περάσουμε τον τίτλο της σελίδας καλώντας την μεθοδο set() η οποία ανήκει στην ViewModel Class τις οποίας οι μέθοδοι set και get ειναι διαθέσιμες στο BaseModel.

```
class ViewModel {
    //dynamically adds a property or method to the ViewModel instance
    public function set($name,$val) {
        $this->$name = $val;
    }
    //returns the requested property value
    public function get($name) {
        if (isset($this->{$name})) {
            return $this->{$name};
        } else {
            return null;
        }
    }
}
```

Εικόνα 18

Αν μια model class κληρονομήσει το BaseModel, αυτομάτως θα έχει πρόσβαση στο ViewModel αντικείμενο μέσω του $this->viewModel. Όπως αναφέραμε νωρίτερα ύπάρχει η μεθοδος set η οποία μας επιτρέπει να προσθέτουμε δυναμικά properties στο viewModel, για παραδειγμα με το $this->viewModel->set('pageTitle', ‘Home’), αποθηκεύει την συμβολοσειρά ‘Home’ στο $this->viewModel->pageTitle.

Γενικά το framework έχει σχεδιαστεί για να κληρονομουν τα models το BaseModel και να δημιουργουν αντικειμενα μορφής viewModel διαθέσιμα για επιστροφή στον controller, ο οποίος μεσα απο την output() θα επιστρεψει στο view τα δεδομένα αυτά.

Οτιδήποτε περαστέι ώς πρωτο όρισμα στην view->output() θα έιναι διαθέσιμο ως $viewModel μεταβλητη στα templates και στα views. Αν το όρισμα μεταβλητής ειναι τύπου ViewModel, τοτε στα templates τα δεδομένα είναι διαθέσιμα μέσω της μεθόδου $viewModel->get('str'), όπου str το όνομα της property που δημιουργήθηκε.

Όπως στο προηγούμενο παράδειγμα αποθηκεύσαμε την συμβολοσειρά 'Home' σε $this->viewModel->pageTitle, έτσι αντίστοιχα με την viewModel->get('PageTitle') θα πέρναμε την συμβολοσειρα στο view.

```php
<?php

class HomeModel extends BaseModel
{
    //data passed to the home index view
    public function index()
    {
        $this->viewModel->set("pageTitle","Ηλεκτρονικό Βιβλία");
        return $this->viewModel;
    }
}

?>

```

Εικόνα 19

```php
<?php foreach ($viewModel->get('comments') as $comment) : ?>
    <div class="comment clearfix">
        <div class="comment-body col-lg-9">
            <h5><a href="" title=""><?php echo $comment['created_at'].'_'.$comment['id'] ?></a></h5>
            <div class="message-sumary">
                <?php echo $comment['comment'] ?>
                <a href="#" class="expander"><small></small></a>
            </div>
        </div>
```

Εικόνα 20

5.4 Παρουσίαση και απεικόνιση πληροφορίας (View)

Ο στόχος που επιτυγχάνεται μεσώ αυτής της πλατφόρμας ειναι ο διαχωρισμός του κωδικα που αφορα την λειτουργία του συστήματος και του business logic (ελεγκτές και υποπρογράμματα που υποστηρίζουν βασικές λειτουργίες του συστήματος), του κώδικα που αφορα την είσοδο, έξοδο και επεξεργασία των δεδομένων, ετσι ώστε να απομονώσουμε απο όλα αυτα τον κώδικα που αφορα την παρουσίαση και απεικόνηση της πληροφορίας(Views). Αυτο σημαίνει οτι τα views θα πρέπει να έχουν λιγοτερο κωδικα εξω απο το context της HTML, XML η οτιδηποτε άλλο χρησιμοποιούμε για τους browsers αυτές τις μέρες.

```php
class View {

    protected $viewFile;

    //establish view location on object creation
    public function __construct($controllerClass, $action) {
        $controllerName = str_replace("Controller", "", $controllerClass);
        $this->viewFile = "views/" . $controllerName . "/" . $action . ".php";
    }

    //output the view
    public function output($viewModel, $template = "maintemplate") {

        $templateFile = "views/".$template.".php";
        //die($templateFile);
        if (file_exists($this->viewFile)) {
            if ($template) {
                //include the full template
                if (file_exists($templateFile)) {
                    require($templateFile);
                } else {
                    require("views/error/badtemplate.php");
                }
            } else {
                //we're not using a template view so just output the method's view directly
                require($this->viewFile);
            }
        } else {
            require("views/error/badview.php");
        }

    }

     /**
       be possible by creating a "setter" method on the View class allowing you to change the viewFile property
      */

    public function setViewFile($_controllerName, $tplFile){
        $this->viewFile = "views/" . $_controllerName . "/" . $tplFile . ".php";
    }
}

?>
```

Εικόνα 21

Στην σχήμα εικόνα 20 βλεπουμε πως δημιουργούμε ενα view που παρουσιάζει τις εγγραφές σχολίων στην διεπαφή, με βάση του πίνακα δεδομένων που έλαβε ο controller απο το αντιστοιχο model και τον επέστρεψε στο view.

Με τη μέθοδο $viewModel->get() επιστρεφουμε τα comments σε μορφή πίνακα και τα με μια foreach αναπαριστουμε μια λίστα με σχόλια τα οποία θα έχουν τη δομή που περιγράφουμε με html, όπως το τίτλο του σχολίου το κυρίως μήνυμα, το μοναδικό id, και οτιδήποτε περιγραφεται στις λειτουργικές απαιτήσεις του χρήστη.

Η δομή που περιγράφει την απεικόνιση της πληροφορίας, κατα κοινή αποδοχή στο web development την ονομάζουμε template, στην επιλογή της δομής αυτής ο κύριος αρμόδιος μηχανισμός είναι η view->output().

Η view->output() είναι μεθοδος της κλάσης view (εικόνα 21). Σχεδόν σε κάθε request ο basecontroller δημιουργεί στιγμιότυπο view, ετσι ώστε να είναι διαθέσιμο στον αντίστοιχο controller που δημιουργήται μεσω του Front Controller μας απο το υποπρόγραμμα loader.

Η view->output() παίρνει ως ορίσματα το αντικείμενο $viewModel για το οποίο έχουμε αναφερθέι εκτενώς σε προηγουμενη ενότητα και το $template που είναι μεταβλητή τύπου string και μεσα απο αυτο επιλέγουμε το view template το οποίο περιγραφει την δομή απεικόνιση της πληροφορίας που θελουμε να επιστρεψουμε
στον χρήστη.

Αυτο που κάνει κυρίως ειναι να εντοπίζει το αρχειο με την html που θελουμε και να το φορτωνει , με προεπιλογή το maintemplate το οποίο ειναι και ο σκελετός του html της εφαρμογής. Σε περίπτωση που το template δεν υπάρχει και αυτο μπορεί να μεταφραστει και ώς σε περιπτωση που η σελίδα που ζητά ο client user δεν υπάρχει, επιστρέφει ενα template που αναφέρει οτι η σελίδα που ζητήθηκε δεν υπάρχει.

ΚΕΦΑΛΑΙΟ 6 Παρουσίαση εφαρμογής ηλεκτρονικό βιβλίο

6.1 Περιγραφή συστήματος

Ο αρχικός σχεδιασμός της διαδικτυακής εφαρμογής Ηλεκτρονικό Βιβλίο πραγματοποιήθηκε με βάση τα παρακάτω:

1. Αρχικά θα πρέπει να υπάρχει στη Βάση Δεδομένων ένας διαχειριστής με όνομα admin και κωδικό admin
2. Τα σχόλια μπορούν να εισάγονται απο οποιονδήποτε χρήστη, χωρις πιστοποίηση πρόσβασης.
3. Τα σχόλια μπορούν να διαγράφονται, να εγγκρίνονται, και να διατηρούνται σε κατάσταση εκκρεμμότητας μόνο απο τον διαχειριστή.
4. Το σύστημα θα πρέπει να παρέχει φόρμα νέου σχολίου στον χρήστη με πεδία:
    1. Ονοματεπώνυμο χρήστη
    2. Επάγγελμα και ασχολία
    3. ηλικία
    4. Το σχόλιο
    5. φύλο
    6. email
    7. τηλέφωνο
    8. Κουμπί αποστολής
5. Το σύστημα θα πρεπει να επεξεργάζεται όλα τα πεδία της φόρμας και αν όλα ειναι έγκυρα να τα αποθηκεύει στη Βάση μαζι με την ημερομηνία εγγραφής και με μοναδικό id. Επιπλέον θα εμφανίζει μηνύματα λάθους δίπλα στα πεδία που είχαν πρόβλημα, περνοντας προς τη φόρμα νέου σχολίου τις τιμές των πεδίων και τα μηνυματα λάθους.

6. Το σύστημα θα παρέχει στον διαχειριστή λίστα με εγγραφές οπου κάθε εγγραφή θα έχει τα παρακάτω:
    1. Τίτλο με την ημερομηνία και ώρα που δημιουργήθηκε μαζί με το μοναδικό id εγγραφής στη βάση.
    2. Το περιεχόμενο του σχολίου
    3. Τα περιεχομενα των υπολοιπων πεδίων όπως Ονοματεπώνυμο, επαγγελμα-ασχολία, ηλικία Διεύθυνση ηλεκτρονικού ταχυδρομείου, τηλέφωνο, φύλο.
7. Το σύστημα θα παρέχει στον διαχειριστή κουμπι αποδοχής σχολίου, κουμπί διαγραφής σχολίου
8. Το σύστημα με την επιλογη αποδοχής σχολίου θα πρεπει να κανει προσθηκη του σχολίου στον τωρινό μήνα
9. Το σύστημα με την επιλογή απόρριψη θα διαγράφει την εγγραφή απο την βάση.
10. Το σύστημα θα πρεπει να παρέχει λίστα με τις εγγραφές σχολίων, με παρουσίαση τα σχόλια των χρηστων του αμέσως προηγούμενου μήνα και του τωρινού μήνα, ενω θα πρεπει να παρέχει επιλογη φιλτραρίσματος για εμφάνιση αποτελεσμάτων συγκεκριμένου μήνα και έτους.

6.2 Περιγραφή βάσης δεδομένων

Μια Βάση Δεδομένων (ΒΔ) είναι ένα σύνολο αρχείων με υψηλό βαθμό οργάνωσης τα οποία είναι συνδεδεμένα μεταξύ τους με λογικές σχέσεις, έτσι ώστε να μπορούν να χρησιμοποιούνται από πολλές εφαρμογές και από πολλούς χρήστες ταυτόχρονα.

Η παρούσα εργασία βασίζεται πάνω σε php και MySql. Περιλαμβάνει μια βάση δεδομένων η οποία περιέχει πίνακες δεδομένων. Μέσα σε κάθε πίνακα δηλώνουμε τον τύπο του κάθε πεδίου (π.χ. char, int, datetime κλπ) και το πρωτεύον κλειδί του (primary key), το οποίο μπορεί να αποτελείται από ένα πεδίο ή και από συνδυασμό περισσοτέρων πεδίων με την προϋπόθεση να είναι μοναδικό (unique).

Σύμφωνα με την περιγραφή συστήματος χρειαζόμαστε δύο πίνακες, τον πίνακα “comments” στον οποίο θα υπάρχουν οι εγγραφές σχολίων του κάθε χρήστη, και ο πίνακας “users” στον οποίο θα ύπαρχουν εγγραφές με τους εξουσιοδοτημένους χρήστες που έχουν ρόλο διαχειριστη.

Εικόνα 22

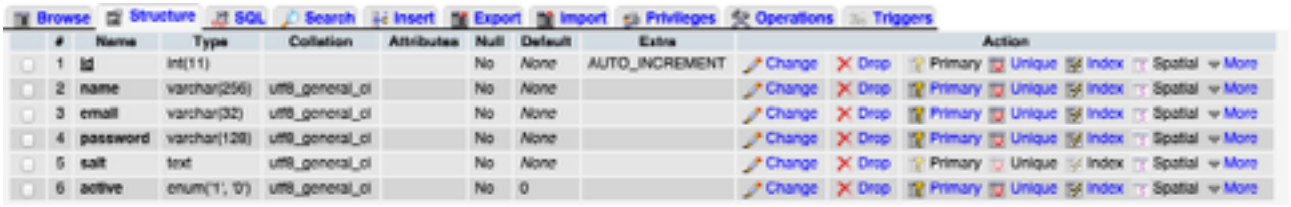
6.2.1 Οντότητες

Ο σχεδιασμός μιας βάσης δεδομένων μπορεί να προκύψει απο την περιγραφή συστηματος, σε αυτη την εφαρμογή μέσα απο την περιγραφή του συστήματος προκύπτουν δύο οντότητες. Η οντότητα σχόλιο, που αποτελείται εκτός απο το ίδιο το κείμενο ενος χρήστη, αλλα και απο έξτρα πεδια εισόδου όπως το ονοματεπώνυμο του χρήστη που το έγραψε, την ηλικία του κλπ. Η οντότητα διαχειριστής που αποτελείται απο το κωδικό πρόσβασης ή και κάποιο email.

Ο πίνακας comments αποτελείται απο δώδεκα πεδία όπως φαινεται στην εικόνα 21, το id που ειναι το primary key και μοναδικό του πίνακα, το is_approved που είναι πεδίου enum, δηλαδή μπορεί να πάρει 2 προκαθορισμένες τιμές σε αυτη τη περίπτωση ‘yes’ και ‘no’, τις created_at και updated_at πεδία τύπου datetime για τις ημερομηνίες.

Τα πεδία title, fullname, email phone, occupation, είναι τύπου varchar με μεγιστη τιμή τους 255 χαρακτήρες και σε αυτα αποθηκεύονται τα αντίστοιχα δεδομένα του τίτλου του σχολίου, το ονοματεπώνυμο του χρήστη, το email του, το τηλέφωνο του και το επαγγελμα του.

Το πεδίο content είναι τύπου text και αυτο σημαινει οτι μπορεί να πάρει χαρακτήρες πολύ μεγαλυτερου μεγέθους απο τα πεδία τύπου varchar(255). Επίσης το πεδίο age που είναι τύπου int καθώς και το πεδίο gender που ομοίως με το is_approved είναι τύπου enum.



εικόνα 22

Ο πίνακας users (εικόνα 22) αποτελείται και αυτος απο το column field id το οποίο έιναι μοναδικό, απο το πεδίο name, τύπου varchar το οποίο είναι διαθέσιμο στο να βαλει ο διαχειριστής το πραγματικο του όνομα αν χρειαστεί, το email του διαχειριστή, απο το password στο οποίο θα καταχωρήται ο κωδικος του χρήστη, το πεδίο salt το οποίο υπάρχει στη περίπτωση που θέλουμε να κρυπτογραφούμε τους κωδικούς του χρήστη, και το πεδίο active το οποίο είναι τύπου enum και με αυτο θετουμε τον χρήστη ενεργο ή όχι.

ΚΕΦΑΛΑΙΟ 7 Τεκμηρίωση εφαρμογής

Εικόνα 23

Στην εικόνα 23 φαίνεται η αρχική σελίδα της εφαρμογής. Το μενού της εφαρμγης αποτελείται απο το Βιβλίο επισκεπτών, την Δημιουργία Σχολίου την Διαχείριση και την Σύνδεση του Διαχειριστή στο σύστημα διαχείρισης σχολίων.

7.1 Δημιουργία Σχολίου

Εικόνα 24

Εδώ βλεπουμε μια φόρμα εισαγωγής σχολίου, που περιγραψαμε στο Κεφαλαιο 5 Ο χρήστης πληκρολογεί τα δεδομένα του, και μολις πατήσει καταχώρηση, και πρωτου φτασει στη βάση δεδομένων, γινεται έλεγχος για data integrity των δεδομένων εισόδου.

Αυτη την εργασία την αναλαμβάνει το υποπρογραμμα validator.php το οποίο βρίσκεται στο libs/ validator.php.

Εικόνα 25

```
function validateform(){

  jQuery('#'+controller+'-form').submit(function() {
    jQuery.ajax({
        type: "POST",
        url: '/'+controller+'/validate',
        data: jQuery('#'+controller+'-form').serialize(),
        success: function(data)
        {
          parsed_data = JSON.parse(data);
          //console.log(parsed_data);
          jQuery.each( parsed_data, function( index, value){
              console.log(index);
              jQuery('#'+index).css({'display' : 'inline-block'}).html(value);
          });
          submitForm();
        }
    });
    return false;
  });
}

function submitform(){

    //console.log(controller);
    //console.log(action);
    var url = "/"+controller+"/"+action; // the script where you handle the form input.
    jQuery.ajax({
        type: "POST",
        url: url,
        data: jQuery('#'+controller+'-form').serialize(), // serializes the form's elements.
        success: function(data)
        {
          jQuery('#status').html(data);
        }
    });
}
```

Εικόνα 26

Στην εικόνα 25 ο χρήστης κάνει ανεπιτυχή αποπειρα καταχώρησης, όπως θα δείτε το υποπρογραμμα validation επιστρέφει δύο μηνύματα λάθους αντίστοιχα στα πεδία που συμβαίνει τι πρόβλημα, στην πρωτη περίπτωση ο χρήστης δεν πληκτρολογεί εγκυρο email format στο πεδίο και στη δεύτερη περιπτωση έχει ξεχάσει να πληκτρολογίσει αριθμό τηλεφώνου του οποίου το πεδίο ειναι υποχρεωτικό. Επίσης το τελικό μήνυμα που επιστρέφει είναι οτι "το σχόλιο σας δεν καταχωρήθηκε". Στη συνέχεια θα περιγράψουμε πως γίνεται ο ελεγχος της φόρμας.

```php
class CommentController extends BaseController
{
    //add to the parent constructor
    public function __construct($action, $urlValues) {
        parent::__construct($action, $urlValues);

        //create the model object
        require("models/comment.php");
        $this->model = new CommentModel();
    }

    //default method
    protected function index()
    {
        $this->view->output($this->model->index());
    }

    protected function validate(){
        $form_data = $_POST;
        $validation_status = $this->model->showValidationReport($form_data);
        print json_encode($validation_status['error_reports']);
    }
```

```php
31      public function showValidationReport(){
32          $form_data = $_POST;
33          $validation = $this->validator->validate($form_data, $type = 'comment');
34          return $validation;
35      }
```

Εικόνα 27

Η αποστολή και ο έλεγχος της φόρμας γίνεται ασύγχρονα, με χρήση ajax (εικόνα 26). Ο λόγος που το κάνουμε αυτο είναι για να αποφύγουμε την επαναφορτωση των αρχείων της εφαρμογής μιας και εμεις χρειαζομαστε καποια strings ως μηνύματα για το αν η αποστολή πηγε καλά σε περιπτωση επιτυχίας, και μηνύματα λάθους σε περίπτωση προβλήματος.

Οταν ο χρήστης κάνει submit, στη περίπτωση μας το url μας ειναι comment/validate, κανουμε request δηλαδη την validate μεθοδο της commentController class, η οποία καλεί τη μέθοδο του αντιστοιχου model CommentModel την showValidationReport (Εικόνα 27), το οποίο CommentModel υιοθετεί την validation Class του υποπρογραμματος validator.php που είναι διαθέσιμο στο BaseModel κατα την κατασκεύη στιγμιοτύπού της ως στιγμιότυπο $this->validator.

Η showValidationReport() πάιρνει όρισμα τα $_POST δεδομένα της φόρμας, και έχει ως δεύτερο όρισμα $type = 'comment', έτσι ώστε όταν θα κληθεί η validate να γνωρίζει απο πριν οτι προκειται να γινει ελεγχος τύπου comment, διοτι όπως θα δουμε η ίδια Validation χρησιμοποιείται και στη περίπτωση εισόδου χρήστη.

Η Validation Class έχει 2 βασικές λειτουργίες η οποίες εκτελούνται με την εξής σειρά:

1. $this->cleanInput($input); και $this->sanitize($input); . Αυτές οι δύο συναρτήσεις εκτελουν το sanitization των εισερχόμενων δεδομένων στη βάση. Sanitization είναι η διαδικασία αφαίρεσης ευαίσθητης πληροφορίας από ένα έγγραφο ή φόρμα έτσι ώστε να είναι διαθέσιμη με ασφάλεια στην εισαγωγή στη βάση δεδομένων.
2. Η $this->validate(), η οποία αποτελεί τον πυρήνα της διαδικασίας, η validate δέχεται ώς όρισμα πίνακα δεδομένων που επιστρέφουν οι cleanInput() και sanitize() και εξετάζει ένα ένα τα δεδομένα που ο τύπος τους αντιστοιχεί σε εναν protected πίνακα που χαρακτηρίζει των τύπο δεδομενων αυτων με τη βοήθεια των regular expressions. Αν δεν ικανοποιήθεί ο κανόνας regex που έχουμε ορίσει το λάθος προστιθεται σε εναν πίνακα λαθών ο οποίος επιστρέφεται και είναι διαθέσιμος στον controller για να περαστεί στο view. Στο τέλος αν ο πίνακας λαθων είναι άδειος σημαίνει οτι τα δεδομένα συμορφώνονται στους κανόνες που έχουμε θέση και επιστρέφει οτι τα δεδομένα είναι διαθέσιμα για εισαγωγή στη βάση.

```
<?php

class Validation {

    protected $data = array();
    protected $error = array(
        'comment' => '',
        'fullname' => '',
        'occupation' => '',
        'age' => '',
        'email' => '',
        'phone' => ''
    );

    protected $login_error = array(
        'email' => '',
        'password' => '',
    );

    protected function sanitize($input) {
        if (is_array($input)) {
            //die('its array');
            foreach($input as $var=>$val) {
                $output[$var] = $this->sanitize($val);
            }
        }
        else {
            if (get_magic_quotes_gpc()) {
                $input = stripslashes($input);
            }
            $input  = $this->cleanInput($input);
            $output = mysql_real_escape_string($input);
        }
        return $output;
    }
```

```
    protected function cleanInput($input) {

      $search = array(
        '@<script[^>]*?>.*?</script>@si',   // Strip out javascript
        '@<[\/\!]*?[^<>]*?>@si',            // Strip out HTML tags
        '@<style[^>]*?>.*?</style>@siU',    // Strip style tags properly
        '@<![\s\S]*?--[ \t\n\r]*>@'         // Strip multi-line comments
      );

        $output = preg_replace($search, '', $input);
        return $output;
    }

    public function validate($input, $type=false) {
        if(!empty($input)){
            //print_r($input);

            $input = $this->cleanInput($input);
            $input = $this->sanitize($input);
            //print_r($input);
            //die;
            if($type === 'comment'){
                foreach ($input as $field => $value) {
                    switch ($field) {
                        case 'comment':
                            if(empty($input[$field]))
                                  $this->error['comment'] = "comment field is
required";
                            break;

                        case 'fullname':
                            if (empty($input[$field])) {
                                $this->error['fullname'] = "fullname field is
required";
                            }else{
                                      if(!preg_match('/^[\p{L}\.\'\- ]+$/u',
$input[$field]))
                                    $this->error['fullname'] = "this is not a
name";
                            }
                            break;

                        case 'occupation':
                            if (empty($input[$field])) {
                                     $this->error['occupation'] = "occupation
field is required";
                            }else{
                                      if(!preg_match('/^[\p{L}\.\'\- ]+$/u',
$input[$field]))
                                    $this->error['occupation'] = "this is not
occupation";
                            }
                            break;

                        case 'age':
                            if (empty($input[$field])) {
                                        $this->error['age'] = "age field is
required";
                            }else{
```

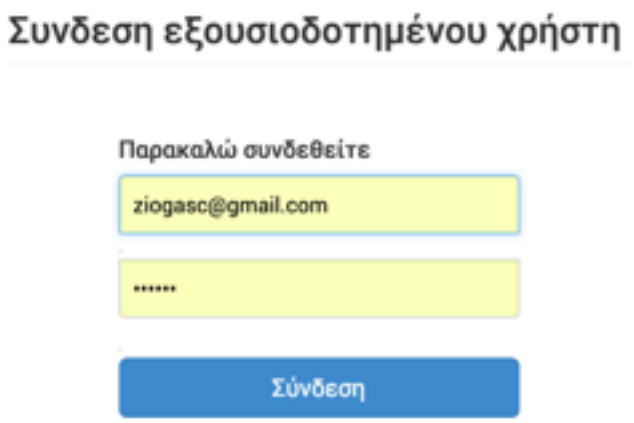
```
                                        if(intval($input[$field]) < 8 ||
intval($input[$field]) > 70)
                                            $this->error[] = "this is not a valid
age";
                            }
                            break;

                        case 'email':
                            if(empty($input[$field])) {
                                    $this->error['email'] = "email field is
required";
                            } else {
                                if(!preg_match("/^[a-zA-Z]\w+(\.\w+)*\@\w+(\.
[0-9a-zA-Z]+)*\.[a-zA-Z]{2,4}$/", $input[$field]))
                                    $this->error['email'] = "this is not a
valid email address";

                            }

                            break;
                        /*
                        case 'gender':
                            if(empty($input[$field]))
                                    $this->error['gender'] = "gender is
required";
                            break;
                        */
                        case 'phone':
                            if(empty($input[$field])) {
                                $this->error['phone'] = "phone field is
required";
                            } else {
                                if(!is_numeric($input[$field]))
                                    $this->error['phone'] = "this is not a
valid phone number";
                            }
                            break;
                    }
                }

                $this->data['error_reports'] = $this->error;
                                $this->data['general_status'] = $this-
>validation_status($this->error);

            } else if ($type === 'login') {
                foreach ($input as $field => $value) {
                    switch ($field) {
                        case 'email':
                            if(empty($input[$field])) {
                                $this->login_error['email'] = "email field is
required";
                            } else {
                                if(!preg_match("/^[a-zA-Z]\w+(\.\w+)*\@\w+(\.
[0-9a-zA-Z]+)*\.[a-zA-Z]{2,4}$/", $input[$field]))
                                    $this->login_error['email'] = "this is
not a valid email address";
                            }
                            break;
                        case 'password':
                            if(empty($input[$field])) {
```

```
                                    $this->login_error['password'] = "password
field is required";
                            } else {
                                if(strlen($input[$field]) < 6 )
                                            $this->login_error['password'] =
"password must be at least 6 characters long";
                            }
                            break;
                    }
                }

                $this->data['error_reports'] = $this->login_error;
                                        $this->data['general_status']  =  $this-
>validation_status($this->login_error);
            }


            $this->data['healthy_data'] = $input;

            return $this->data; // form is validated and sanitized

        }else{

            if($type === 'comment'){
                $this->error[] = 'post array is completely empty';
                return $this->error;
            }

            if($type === 'login'){
                $this->login_error[] = 'post array is completely empty';
                return $this->login_error;
            }
        }
    }

    protected function validation_status($args){

        if(!array_filter($args)){
            return true;
        } else {
            return false;
        }
    }

}
?>
```

/libs/validator.php

7.2 Συνδεση εξουσιοδοτημένου χρήστη

Όταν ο χρήστης κάνει submit την φόρμα σύνδεσης, αναλαμβάνει ο login controller ο οποίος θα καλέσει την μέθοδο doLogin.



Εικόνα 28

Η doLogin με τη βοήθεια μεταβλητής $post_data, αποθηκεύει την post array της φόρμας για να είναι διαθέσιμη προς επεξεργασία. Έπειτα καλεί την $this->model->setSession($post_data), η οποία είναι μέθοδος του login model η οποία κάνει δύο πράγματα, α. περνάει τα post data της φόρμας από validation, και αν η validation επιστρέψει ότι όλα πήγαν καλά συνεχίζουμε με β. την $this->_checkCredentials().

Η CheckCredentials είναι και αυτή μέθοδος του LoginModel, μέριμνα της είναι να κάνει query την βάση για

```
    protected function _checkCredentials($flag='')
    {
            $stmt = $this->_db->prepare('SELECT * FROM users WHERE
email=?');
        $stmt->execute(array($this->_email));

        if ($stmt->rowCount() > 0 && $flag != 'register') {
            $user = $stmt->fetch(PDO::FETCH_ASSOC);
             /**$submitted_pass = $user['salt'].sha1($this->_password);
**/
            if ($this->_password == $user['password']) {
                return $user;
            }
        } elseif ($stmt->rowCount() > 0 && $flag === 'register') {
            // email found
            return true;
        } else{
            // email not found
            return false;
        }
    }
```

checkCredentials()

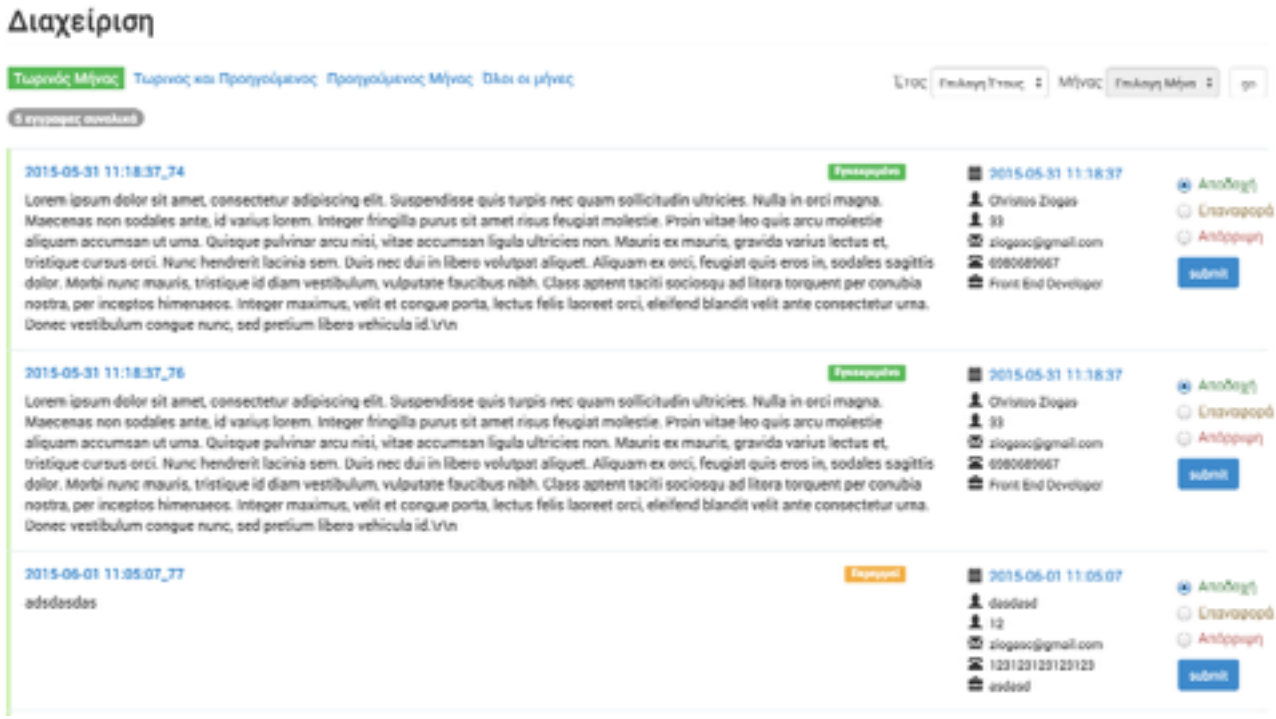
το αν υπάρχει email καταχωρημένο στον πίνακα users και αν υπάρχει, να κάνει σύγκριση με το εισαγομενο κωδικό πρόσβασης με αυτό που είναι ήδη καταχωρημένο στη βάση. Αν η _checkCredentials επιστρέψει false αυτό θα σημαίνει ότι δεν βρέθηκε καταχώρηση με αυτά τα στοιχεία, διαφορετικά θα επιστρέψει $user, και αυτομάτως θα σημαίνει ότι βρέθηκε ο χρηστης.

```
   public function setSession($post_data)
    {
          $validation = $this->validator->validate($post_data, $type =
'login');

        if(count(array_filter($validation['error_reports'])) == 0){

            $this->_email = $validation['healthy_data']['email'];
                        $this->_password = $validation['healthy_data']
['password'];


            $user = $this->_checkCredentials();
            if ($user) {
                //die('found');
                       $this->_user = $user; // store it so it can be
accessed later
                $_SESSION['user_id'] = $user['id'];
                //return $user['id'];
                return true;
            }
            return false;

        } else {
            //print json_encode($validation['error_reports']);
            //return $validation['error_reports'];
            return false;
    }
  }
```

setSession()

Στη συνέχεια η setSession ελέγχει αν η _checkCredentials επέστρεψε χρήστη, και δημιουργεί session με τον χρηστη.

7.3 Διαχείριση Σχολίων



Εικόνα 29

Στην εικόνα 29 βλεπουμε την διεπαφή της διαχείρισης σχολίων απο τον διαχειριστή. Είναι μια διεπαφη απο την οποία βλεπουμε όλα τα σχολια που εχουν περαστεί στη βάση. Ο διαχειριστής μπορέι να τα ταξινομήσει με βάση τον τωρινο μήνα, τον τωρινό και αμέσως προηγούμενο μήνα, τον προηγουμενο μήνα, και χωρίς κάποιο φιλτρο δηλαδη όλους τους μήνες. Επίσης μπορεί να κάνει έυρεση με βαση το έτος και τον συγκεκριμενο μήνα του έτους.

Σε κάθε σχόλιο ο χρήστης μπορέι θέσει το σχόλιο σε κατασταση αποδοχής, επαναφοράς, και απορριψης(απορριψη θα σημαίνει και διαγραφή).

7.3.1 Ταξινόμιση

Κάθε κουμπί ταξινόμισης ουσιαστικά αντιστοιχεί με μια μεθοδο του adminController, για παράδειγμα οταν ο διαχειριστής επιλέξει προβολή τωρινού και προηγούμενου μήνα το url θα έιναι admin/currentprevmonths,το request αυτό αντιστοιχεί με την μέθοδο currentprevmonths() του adminController,

Εικόνα 30

```
    protected function currentprevmonths(){
        $this->view->setViewFile('Admin', 'index');
```

```
        $this->view->output($this->model->list_current_prev_months());
    }
```

η οποία μεθοδος με τη σειρά της καλεί την αντίστοιχη μέθοδο του adminModel την list_current_prev_months().

7.3.2 Ο χρήστης επιλέγει συγκεκριμένο έτος και μήνα

Εικονα 31

Τα διαθέσιμα έτη εμφανίζονται κατα την φόρτωση της σελίδας, είναι παράγωγα της συνάρτησης adminModel->list_available_years η οποία εκτελείται μέσα απο την κατασκευή στιγμιοτύπου adminModel και επιστρέφονται στο view απο την αρχή.

Όταν ο χρήστης επιλέξει ένα έτος, τοτε στο view και με χρήση javascript εκτελείται η populateMonths().

Η populateMonths είναι μια ajax function ή οποία ασύγχρονα πραγματοποιεί request στην /admin/ listmonths μεθοδο του controller adminController, η οποία στη συνέχεια καλεί την list_available_months του adminModel.

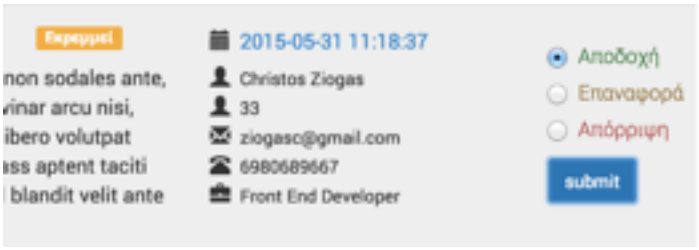
```
    public function list_current_prev_months()
    {
                    $stmt = $this->_db->query('SELECT * FROM comments WHERE
MONTH(created_at) > MONTH(CURDATE()-INTERVAL 2 MONTH)');
        // TODO: Need
        $results = $stmt->fetchAll(PDO::FETCH_ASSOC);
        $num_of_results = count($results);
        $month = $this->monthname($results[0]['created_at']);

        $this->viewModel->set("current_month", $month);
        $this->viewModel->set("comments", $results);
        $this->viewModel->set("num_of_results", $num_of_results);


        return $this->viewModel;
   }
```

list_current_prev_months()

Η list_current_prev_moths() τρέχει ένα select mysql query στη βάση, το οποίο ψάχνει μέσα στον πίνακα comments όλες τις καταχωρίσεις που έχουν γίνει στον τωρινό μήνα και στον αμέσως προηγούμενο και τις επιστρέφει.

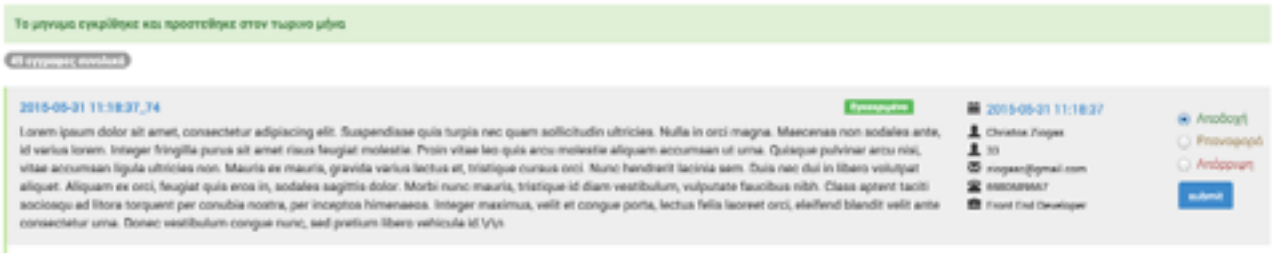
Ομοίως η ίδια διαδικασία και λογική ακολουθείται και στα υπολοιπα φίλτρα ταξινόμισης εκτος απο αυτο της επιλογής συγκεκριμένου έτους και μήνα που ακολουθήται μια τεχνικά ασύγχρονη διαδικασία με χρήση ajax.

7.3.3 Έγκριση σχολίου και προσθήκη στον τωρινό μήνα



Εικόνα 32

Ο Διαχειριστής έχει την δυνατότητα να αποδεχτεί το μήνυμα, να το επαναφέρει και να το απορρίψει. Στα δεξιά κάθε καταχώρησης (εικόνα 32) υπάρχει μια φόρμα απο radiobuttons, επιλέγοντας "αποδοχή" το σύστημα κάνει προσθήκη του σχολίου στον τωρινό μήνα και το θέτει σε κατάστασή εγκρισης (Εικόνα 33).



Εικόνα 33

Ο διαχειριστής επίσης έχει την δυνατότητα να επαναφέρει ένα εγκεκριμένο μήνυμα σε κατάσταση "Εκρεμμεί", στην εικόνα 34 και 35 θα παρατηρήσετε οτι υπάρχει διαθέσημο ένα label με τη τωρινή κατάσταση του μηνύματος. Με την επιλογή της "Απόρριψης" το σύστημα διαγράφει οριστικά το μήνυμα απο τη βάση δεδομένων.

Εικόνα 36

Στην εικόνα 36 βλεπουμε την μέθοδο action του admin Controller ή οποία δέχεται δεδομένα $_POST της φορμας εγγκρισης-προσθήκης-διαγραφής σχολίου, και αναλόγως την ενέργεια τοτ διαχειριστή καλείται δυναμικά η αντιστοιχη μέθοδος του admin Model.

ΚΕΦΑΛΑΙΟ 8 Συμπεράσματα

Αποτέλεσμα της πτυχιακής μου εργασίας ήταν ή δημιουργία ενός PHP MVC framework και η υλοποίηση της εφαρμογής ηλεκτρονικού βιβλίου πάνω σε αυτό. Είχα την ευκαιρία να εμπλουτίσω τις γνώσεις μου πάνω στο σχεδιαστικό πρότυπο MVC καθώς και να εφαρμόσω αρχές αντικειμενοστραφή προγραμματισμού σε γλώσσα php, επίσης χρησιμοποίησα σχεσιακή βάση δεδομένων mysql μέ χρήση PDO API. Κάνοντας χρήση όλων των παραπάνω με βοήθησε να κατανοήσω και να έχω τη δυνατότητα να χρησιμοποιήσω στο μέλλον τη δομή και τη λειτουργία ενός MVC framework σε οποιαδήποτε γλωσσα προγραμματισμού.