ΑΛΕΞΑΝΔΡΕΙΟ Τ.Ε.Ι. ΘΕΣΣΑΛΟΝΙΚΗΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ Τ.Ε.

Τμήμα Μηχανικών Πληροφορικής ΑΤΕΙΘ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Ανάπτυξη Ολοκληρωμένου Πληροφοριακού Συστήματος Διαχείρισης Ηλεκτρονικού Συνεδρίου

Του φοιτητή

Γιάγκου Χρήστου

Αρ. Μητρώου: 113712

Επιβλέπων καθηγητής

Κέρστιν Σιάκα

Θεσσαλονίκη 2018

ΠΡΟΛΟΓΟΣ

Το παρόν κείμενο αποτελεί το γραπτό κομμάτι της πτυχιακής εργασίας με τίτλο Ανάπτυξη ολοκληρωμένου Πληροφοριακού συστήματος διαχείρισης ηλεκτρονικού συνεδρίου.

Η συγκριμένη πτυχιακή εργασία στοχεύει στο σχεδιασμό και την ανάπτυξη μίας ιστοσελίδας ειδικού ενδιαφέροντος, η οποία θα προσφέρει στους χρήστες μεταξύ άλλων, ενημέρωση για ένα συγκεκριμένο συνέδριο, εγγραφή σε αυτό, κατάθεση μιας εργασίας προς παρουσίαση στο συνέδριο, έλεγχος εργασιών και άλλες δυνατότητες οι οποίες θα αναλυθούν παρακάτω. Θα μελετηθούν τα βήματα σχεδιασμού του ιστοτόπου, όπως για παράδειγμα οι πρωταρχικές ενέργειες για την επιλογή των κατάλληλων εργαλείων αλλά και η ανάλυση του συστήματος.

ΠΕΡΙΛΗΨΗ

Το θέμα της παρούσας πτυχιακής εργασίας εντάσσεται στο αντικείμενο της ανάπτυξης ενός ολοκληρωμένου πληροφοριακού συστήματος διαχείρισης συνεδρίου.

Στα πλαίσια της πτυχιακής εργασίας, το σύστημα που θα αναλυθεί και αναπτυχθεί θα επιτρέπει στους χρήστες του να δημιουργήσουν και να διαχειρίζονται οργανώσεις συνεδρίων, να αναθέτουν ρόλους, όπως αυτού του κριτή σε αντίστοιχες δημοσιεύσεις, να πραγματοποιούνται εγγραφές στα διάφορα συνέδρια, παρουσιάσεις εργασιών, αλλά ακόμη δίνεται και η δυνατότητα παροχής ενημερωτικού υλικού στους χρήστες αυτής.

Τελικός στόχος της παρούσας διαδικτυακής εφαρμογής είναι να δοθεί η δυνατότητα στους χρήστες της να περιηγηθούν στην εφαρμογή, είτε ως απλοί χρήστες, είτε ως διαχειριστές του συστήματος με στόχο την εκπόνηση συγκεκριμένων καθηκόντων. Ως διαχειριστής, δίνεται η δυνατότητα στον χρήστη να έχει τον πλήρη έλεγχο της εφαρμογής, όπως είναι η πλήρη διαχείριση συνεδρίων, η διοργάνωση αυτών, η ανάθεση ρόλων, δημιουργία γεγονότων, κ.α. Ως χρήστης, δίνεται η δυνατότητα ενημέρωσης για τα διάφορα συνέδρια και του υλικού αυτών, αλλά ακόμα και η δυνατότητα εγγραφής και συμμετοχής με την καταβολή αντίστοιχης δημοσίευσης.

Τεχνολογικά, το τελικό προϊόν είναι ένα Web application όπου στο backend χρησιμοποιήθηκε η Java μαζί με το Spring Framework για την κατασκευή RESTful APIs. Για την διαχείριση των δεδομένων χρησιμοποιήθηκε η MySQL/H2. Τέλος, στο client της εφαρμογής, χρησιμοποιήθηκε Angular/TypeScript μαζί με άλλες third-party βιβλιοθήκες οι οποίες συνέβαλαν στην ολοκλήρωσή και σχεδίαση αυτής (όπως Bootstrap).

ABSTRACT

The subject of this final year thesis is part of the development of an integrated information management system for a conference.

In this thesis, the system that we will analyze and develop, will allow its users to create and manage conference organizations, assign roles, such as reviewer of assigned publications, register at conferences, presentations, but it is also possible to provide other information to its users.

The ultimate goal of this web application is to allow its users to navigate the application, either as a single user or as a system administrator and to perform specific tasks. As an administrator, the user has the ability to completely control the application, such as full conference management, including the organizing conferences, assigning roles, creating events, and more. As a user, it is possible to be informed about various conferences, but also to have the ability to register and participate by submitting a corresponding publication(paper).

Technologically, the end product is a Web application where the backend uses Java along with the Spring Framework to build RESTful APIs. MySQL/H2 was used to manage the data. Finally, in the application's client, Angular / TypeScript was used along with other third-party libraries that contributed to its completion and design (such as Bootstrap).

ΕΥΧΑΡΙΣΤΙΕΣ

Αρχικά, θα ήθελα να ευχαριστήσω θερμά την καθηγήτρια Κέρστιν Σιάκα για την εμπιστοσύνη που μου έδειξε αναθέτοντας την παρούσα πτυχιακή εργασία, για την εξαιρετική συνεργασία και την πολύτιμη συμβολή της στην ολοκλήρωση αυτής.

Θέλω επίσης να ευχαριστήσω όλους του δικούς μου ανθρώπους που μου στάθηκαν αυτό το διάστημα, με στήριξαν με υπομονή και κατανόηση κατά την διάρκεια της φοίτησής μου και με βοήθησαν να πετύχω τους στόχους μου.

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΕΦΑΛΑΙΟ 1 – ΤΟ ΔΙΑΔΙΚΤΥΟ

ΕΙΣΑΓΩΓΗ

Το Διαδίκτυο (Internet) είναι το παγκόσμιο σύστημα διασυνδεδεμένων δικτύων υπολογιστών, οι οποίοι χρησιμοποιούν μία καθιερωμένη ομάδα πρωτοκόλλων, η οποία συχνά αποκαλείται "TCP/IP" (αν και αυτή δεν χρησιμοποιείται από όλες τις υπηρεσίες του Διαδικτύου), για να εξυπηρετεί εκατομμύρια χρηστών καθημερινά σε ολόκληρο τον κόσμο. Οι διασυνδεδεμένοι ηλεκτρονικοί υπολογιστές ανά τον κόσμο, οι οποίοι βρίσκονται σε ένα κοινό δίκτυο επικοινωνίας, ανταλλάσσουν μηνύματα (πακέτα) με τη χρήση διαφόρων πρωτοκόλλων (τυποποιημένοι κανόνες επικοινωνίας), τα οποία υλοποιούνται σε επίπεδο υλικού και λογισμικού. Το κοινό αυτό δίκτυο καλείται Διαδίκτυο. Το Διαδίκτυο (Internet) είναι ένα επικοινωνιακό δίκτυο που επιτρέπει την ανταλλαγή δεδομένων μεταξύ οποιουδήποτε διασυνδεδεμένου υπολογιστή. Η τεχνολογία του είναι κυρίως βασισμένη στην διασύνδεση επιμέρους δικτύων ανά τον κόσμο και πολυάριθμα πρωτόκολλα επικοινωνίας. Στην πιο εξειδικευμένη και περισσότερο χρησιμοποιούμενη μορφή του, με τον όρο Διαδίκτυο, περιγράφεται το παγκόσμιο πλέγμα διασυνδεδεμένων υπολογιστών και των υπηρεσιών και πληροφοριών που παρέχει στους χρήστες του. Το Διαδίκτυο χρησιμοποιεί μεταγωγή πακέτων και τη στοίβα πρωτοκόλλων. Σήμερα, ο όρος διαδίκτυο κατέληξε στο να αναφέρεται στο παγκόσμιο αυτό δίκτυο. Για να ξεχωρίζει, το παγκόσμιο αυτό δίκτυο γράφεται με κεφαλαίο το αρχικό "Δ". Η τεχνική της διασύνδεσης δικτύων μέσω μεταγωγής πακέτων και της στοίβας πρωτοκόλλων ονομάζεται Διαδικτύωση.

ΥΠΟΚΕΦΑΛΑΙΟ 1.1 – Η ΙΣΤΟΡΙΑ ΤΟΥ ΔΙΑΔΙΚΤΥΟΥ

Το σημερινό διαδίκτυο αποτελεί την εξέλιξη του δικτύου ARPANET που άρχισε να αναπτύσσεται, σε ένα πειραματικό επίπεδο, στις ΗΠΑ στα τέλη της δεκαετίας του ’60.

1. Δεκαετία '60: Πρώτα Πειράματα

Το δίκτυο ARPANET δημιουργείται το 1969 από τα πρώτα πειράματα ερευνητών για την διασύνδεση απομακρυσμένων υπολογιστών μεταξύ τους στα πανεπιστήμια των Η.Π.Α. Χρηματοδοτήθηκε από το πρόγραμμα ARPA
(Advanced Research Project Agency) του Υπουργείου Άμυνας και σκοπός του πειράματος αυτού ήταν η αξιόπιστη σύνδεση του υπουργείου με στρατιωτικούς ερευνητικούς οργανισμούς.
Αρχικά το πρόγραμμα αποσκοπούσε σε μια νέα τεχνολογία για την εποχή εκείνη γνωστή σαν μεταγωγή πακέτων (Packet Switching) στην οποία πειραματιζόντουσαν. Η τεχνολογία αυτή είχε σχεδιαστεί για την μετάδοση δεδομένων τα οποία χωρίζονταν σε πακέτα ωστέ να μπορούν να μοιραστούν πολλοί χρήστες την ίδια επικοινωνιακή γραμμή. Στο τέλος της διαδικασίας αυτής γίνεται η επανασύνθεση του πακέτου σε δεδομένα για να μπορεί ο τελικός χρήστης να τα χρησιμοποιήσει.

2. Δεκαετία '70: Τα πρώτα βήματα των συνδέσεων

Με σκοπό να να ξεπεραστούν διαφορετικοί τρόποι για την μεταγωγή πακέτων που χρησιμοποιούσαν έως τότε τα δίκτυα, ξεκίνησε το 1973 το ερευνητικό πρόγραμμα με την ονομασία Internetting Project (Πρόγραμμα Διαδικτύωσης). Στόχος είναι να μπορούν ανόμοια δίκτυα να επικοινωνούν μεταξύ τους και η μεταγωγή των δεδομένων μεταξύ αυτών να είναι ομοιόμορφη. Αποτέλεσμα της έρευνας αυτής ήταν η εμφάνιση μιας νέας τεχνικής με την ονομασία Internet Protocol (IP) (Πρωτόκολλο Διαδικτύωσης) από την οποία και πήρε μετά το όνομά του το Internet (Διαδίκτυο). Με την νέα αυτή τεχνική συνδέονται μεταξύ τους

διαφορετικά δίκτυα μέσω ενός κοινού πρωτοκόλλου IP και έτσι αποτελούν ένα διαδίκτυο.
Όλοι οι υπολογιστές που ανήκουν σε ένα δίκτυο IP μπορούν να επικοινωνούν με οποιονδήποτε άλλον του διαδικτύου, γιατί είναι ισοδύναμοι. Παράλληλα, σχεδιάζεται για τον έλεγχο της μετάδοσης των δεδομένων μια νέα τεχνική με το όνομα Transmission Control Protocol (TCP) (Πρωτόκολλο Ελέγχου Μετάδοσης). Τότε γεννιούνται και οι προδιαγραφές για την μεταφορά αρχείων μεταξύ υπολογιστών (FTP) και για τον Ηλεκτρονικό Ταχυδρομείο (Email).

3. Δεκαετία '80: Ακαδημαϊκή Κοινότητα & Παγκόσμιο Διαδίκτυο

Το 1983 το Υπουργείο Άμυνας των Η.Π.Α αναγνωρίζει τον συνδυασμό των πρωτοκόλλων TCP και IP ως ένα (TCP/IP).Η γρήγορη εξάπλωση της δικτύωσης των υπολογιστών οφείλεται στην εμφάνιση ενός λειτουργικού συστήματος το Berkeley UNIX, στο οποίο περιλαμβάνεται το πρωτόκολλο TCP/IP.Την ίδια χρονιά το ARPANET το οποίο έχει επιβαρυνθεί, λόγω της πληθώρας των υπολογιστών που έχουν συνδεθεί σε αυτό από πανεπιστήμια, διασπάται σε δύο τμήματα στο MILNET το οποίο αφορά στρατιωτικές επικοινωνίες και στο νέο ARPANET που χρησιμοποιείται αποκλειστικά για την έρευνα στην δικτύωση από την πανεπιστημιακή κοινότητα. Το 1985 το πρωτόκολλο TCP/IP χρησιμοποιείται από το National Science Foundation (NSF) δημιουργώντας ένα δικό του γρήγορο δίκτυο το οποίο ονομάστηκε NSFNET. Σκοπός του NSFNET είναι να συνδέονται,
τόσο με την επιστημονική κοινότητα όσο και μεταξύ τους, πέντε κέντρα υπερ-υπολογιστών.
Προς το τέλος της δεκαετίας του '80 όλο και περισσότερες χώρες προστίθενται στο δίκτυο NSFNET και πολλά πανεπιστήμια και οργανισμοί δημιουργούν τα δικά τους δίκτυα, τα οποία συνδέονται στο παγκόσμιο δίκτυο το οποίο εξαπλώνεται και γίνεται παγκοσμίως γνωστό ως Internet. Αυτό έχει ως αποτέλεσμα το ARPANET να καταργηθεί το 1990.

4. Δεκαετία '90: Το δίκτυο γίνεται παγκόσμιο

Ανάμεσα στις χώρες που συνδέθηκαν στο NSFNET το 1990 βρίσκεται και η Ελλάδα.

Το 1993 παρουσιάζεται από το εργαστήριο του CERN στην Ελβετία το World Wide Web (WWW) ένας παγκόσμιος ιστός. Το WWW είναι ένα σύστημα με το οποίο μπορεί κάποιος να περιηγηθεί στο Internet χρησιμοποιώντας απλά το ποντίκι του υπολογιστή του και να παρακολουθήσει ή να παρουσιάσει διασυνδεδεμένες πληροφορίες οι οποίες είναι σε μορφή πολυμέσων (multimedia). Παράλληλα, οποιοσδήποτε διέθετε PC και Modem θα μπορούσε να συνδεθεί στο Internet, αφού άρχισαν να εμφανίζονται τα εμπορικά δίκτυα που ανήκουν σε εταιρίες παροχής υπηρεσιών Internet (Internet Service Providers-ISP). Αυτό είχε ως αποτέλεσμα το 1995, το NSFNET να καταργηθεί και το φορτίο του να μεταφερθεί σε εμπορικά δίκτυα.

ΥΠΟΚΕΦΑΛΑΙΟ 1.2 – ΙΣΤΟΣΕΛΙΔΕΣ

Ιστοσελίδα είναι ένα σύνολο σελίδων οι οποίες είναι ενωμένες μεταξύ τους και περιέχουν πληροφορίες τόσο σε μορφή ανάγνωσης όσο ήχου και εικόνας. Οι ιστοσελίδες χωρίζονται σε δύο κατηγορίες τις στατικές (static website) και τις δυναμικές (dynamic website)

1. Στατικές ιστοσελίδες

Οι στατικές ιστοσελίδες αποτελούν την πρώτη μορφή ιστοσελίδας που έκανε την εμφάνιση της στο διαδίκτυο και ουσιαστικά είναι απλά έγγραφα σε ηλεκτρονική μορφή. Ο όρος «στατική» δεν σημαίνει ότι η ιστοσελίδα δεν μπορεί να περιέχει κείμενα, φωτογραφίες, υπερσύνδεσμους, ή άλλα κινούμενα γραφικά αλλά το ότι το περιεχόμενο τις ιστοσελίδας δεν αλλάζει εύκολα. Για να γίνει αυτό απαιτείται εμπειρία στο προγραμματισμό ιστοσελίδων και εξειδικευμένες γνώσεις σε γλώσσες προγραμματισμού όπως HTML, CSS, JavaScript και άλλες τεχνολογίες.
Τέτοιες ιστοσελίδες προτείνονται σε περιπτώσεις που επιχειρήσεις ή ιδιώτες θέλουν την παρουσία τους στο διαδίκτυο με εύκολο και οικονομικό τρόπο. Αυτό γιατί, οι στατικές ιστοσελίδες δημιουργήθηκαν για διαδικτυακούς τόπους με μικρό περιεχόμενο που δεν απαιτεί συνεχή ενημέρωση-ανανέωση.

2. Δυναμικές ιστοσελίδες

Οι δυναμικές ιστοσελίδες από άποψη εμφάνισης και περιεχομένου, σε πολλές περιπτώσεις, δεν έχουν τεράστιες διαφορές με τις στατικές. Η κυριότερη διαφορά ανάμεσα τους είναι ότι οι δυναμικές ιστοσελίδες δεν είναι απλά ηλεκτρονικά έγγραφα αλλά πρόκειται για μια εφαρμογή (πρόγραμμα).
Συγκεκριμένα, οι δυναμικές ιστοσελίδες, ανάλογα με τις ανάγκες του χρήστη, αντλούν το περιεχόμενο τους και αποθηκεύουν πληροφορίες χρησιμοποιώντας μια βάση δεδομένων (database) η οποία επιτρέπει στο χρήστη την εύκολη διαχείριση του περιεχομένου, το οποίο είναι συνδεδεμένο αυτόματα με τον ‘’μηχανισμό‘’ της ιστοσελίδας. Η διαχείριση του περιεχομένου θα πρέπει να είναι ασφαλής, για το λόγο αυτό προστατεύεται με κωδικό πρόσβασης (password) για να μην υπάρχει δυνατότητα να εισέλθουν σε αυτή οι επισκέπτες της ιστοσελίδας.

Ο χρήστης (administrator) διαχειρίζεται το περιεχόμενο της δυναμικής ιστοσελίδας μέσω του εύχρηστου μηχανισμού ‘’CMS (Content Management System)’’ , από τον οποίο η επεξεργασία του περιεχομένου γίνεται εύκολα ακόμα και από κάποιον αρχάριο χωρίς εξειδικευμένες γνώσεις σε γλώσσες προγραμματισμού.

ΥΠΟΚΕΦΑΛΑΙΟ 1.3 – DOMAIN NAME

Το DomainName (όνομα τομέα) ή αλλιώς όνομα δικτύου είναι το όνομα που επιλέγουμε για να συνδεθούμε σε μια ιστοσελίδα η οποία είναι ανεβασμένη στο διαδίκτυο, μέσω κάποιου φυλλομετρητή (π.χ. Internet Explorer, Google Chrome, Firefox). Η λέξη-ονομασία αυτή συνοδεύεται από μια κατάληξη η οποία απεικονίζει την γεωγραφική ‘’περιοχή’’ του δικτύου που ανήκει ή την ιδιότητα της ιστοσελίδας. Για παράδειγμα το όνομα domainname.**gr** ανήκει στην Ελλάδα, το domainname.**com** αντιστοιχεί στον εμπορικό τομέα από την λέξη commercial ενώ η κατάληξη domainname**.eu** δραστηριοποιείται στον ευρωπαϊκό χώρο. Έτσι οι χρήστες γράφοντας μια διεύθυνση της μορφής http://www.domainname.gr ή .com ή .eu θα έχουν την δυνατότητα να επισκεφθούν την ιστοσελίδα που επιθυμούν.
Τέλος, βασική προϋπόθεση για να αποκτήσει και να μπορεί κάποιος να χρησιμοποιήσει ένα domainname είναι να το καταχωρίσει-καταγράψει σε έναν κατάλογο ονομάτων γνωστοί και ως domainname registers. Για το σκοπό αυτό υπάρχουν πολλές εταιρίες γνωστές ως καταχωρητές (registers) που αναλαμβάνουν να καταχωρήσουν ένα domainname.

ΥΠΟΚΕΦΑΛΑΙΟ 1.4 – WEB HOSTING

Το Web Hosting (φιλοξενία ιστοσελίδων) από πολλές εταιρίες εμφανίστηκε στις αρχές της δεκαετίας του ’90, γιατί άρχισε να αυξάνεται ολοένα και περισσότερο η ανάγκη για διαδικτυακή παρουσία. Η μεγαλύτερη άνθιση όμως για φιλοξενία ιστοσελίδων εμφανίστηκε την δεκαετία του 2000, όπου όλο και περισσότερες εταιρίες αλλά και ιδιώτες ήθελαν οι ιστοσελίδες τους να έχουν μια θέση στο διαδίκτυο.

Η φιλοξενία ιστοσελίδων λοιπόν, είναι μια διαδικτυακή υπηρεσία που δίνει τη δυνατότητα σε ιδιώτες να αναρτούν τις ιστοσελίδες τους στο διαδίκτυο. Η υπηρεσία webhosting παρέχει την δυνατότητα ανάρτησης των ιστοσελίδων στο διαδίκτυο χωρίς να επιβαρύνονται οι κάτοχοι των ιστοσελίδων με το κόστος των εξοπλισμών, όπως για παράδειγμα κάποιον εξυπηρετητή ή να χρειάζεται μεγάλος αριθμός συνδέσεων και εύρος σύνδεσης (bandwidth).

Ο ιδιοκτήτης της ιστοσελίδας νοικιάζει ένα χώρο στο διαδίκτυο σε κάποιους διακομιστές (υπολογιστές) και ανεβάζει τα αρχεία του ακόμα και την ηλεκτρονική αλληλογραφία του, μέσω προγράμματος (FTPclient), τα οποία προσφέρονται στους επισκέπτες μέσω ασφαλών δικτύων. Οι ιδιοκτήτες μπορούν να διαχειρίζονται τις ιστοσελίδες από τους περιηγητές τους (browser) μέσω κάποιου πίνακα ελέγχου (controlpanel) ή μέσω προγραμμάτων για απομακρυσμένη σύνδεση.

ΥΠΟΚΕΦΑΛΑΙΟ 1.5 – WEB BROWSER

Οι εξυπηρετητές ή αλλιώς διακομιστές διαδικτύου είναι υπολογιστές οι οποίοι παρέχουν υπηρεσίες σε ιστοσελίδες ή σε πελάτες, γνωστούς ως (clients) οι οποίοι μπορούν και φορτώνουν σε έναν υπολογιστή ή συνδέονται μέσω διαδικτύου. Κάθε εξυπηρετητής διαθέτει μια διεύθυνση IP και πιθανόν ένα όνομα τομέα. Κάθε υπολογιστής μπορεί να μετατραπεί σε εξυπηρετητή διαδικτύου με την εγκατάσταση ενός λογισμικού εξυπηρετητή και τη σύνδεση αυτού στο Internet.

Στα τέλη του 1980, κάνει την εμφάνιση της ο Web Browser στον Παγκόσμιο Ιστό. Όμως ο πρώτος Web Browser εμφανίστηκε το 1993 με γραφικό περιβάλλον από την NCSA ο Mosaic.

Με τον όρο φυλλομετρητής ιστού, ή Web Browser, νοείται ένα λογισμικό, ένα πρόγραμμα που είναι σε θέση αποκωδικοποιεί το περιεχόμενο των ιστοσελίδων και που μας φέρνει σε επικοινωνία με τους εξυπηρετητές ιστού (Web Servers) μέσω του πρωτοκόλλου HTTP. Με τον φυλλομετρητή ιστού έχουμε μία αλληλεπίδραση με εικόνες, βίντεο, κείμενα, παιχνίδια και διάφορες άλλες πληροφορίες που βρίσκονται αναρτημένες στην ιστοσελίδα ενός ιστοτόπου στον Παγκόσμιο ιστό ή σ’ ένα τοπικό δίκτυο.

Αξιοσημείωτο είναι πως οι εικόνες και τα κείμενα μπορεί να περιλαμβάνουν υπερσυνδέσμους που οδηγούν σε άλλες ιστοσελίδες του ίδιου ή διαφορετικού ιστοτόπου. Με τη βοήθεια του Web Browser δίνεται στον χρήστη η γρήγορη και εύκολη προσβασιμότητα του σε πληθώρα πληροφοριών σε διάφορες ιστοσελίδες και ιστοτόπους με την χρήση των υπερσυνδέσμων.

Οι πιο διαδεδομένοι και χρησιμοποιημένοι browsers είναι οι:

- Windows Internet Explorer
- Google Chrome
- Mozilla Firefox
- Apple Safari
- Opera

Τέλος, να προστεθεί ότι οι φυλλομετρητές χρησιμοποιούν τη γλώσσα μορφοποίησης HTML μέσω της οποίας γίνεται η προβολή κάθε ιστοσελίδας. Για τον λόγο αυτό η εμφάνιση καθεμιάς μπορεί να διαφέρει από φυλλομετρητή σε φυλλομετρητή.

ΚΕΦΑΛΑΙΟ 2 – ΣΥΣΤΗΜΑΤΑ ΔΙΑΧΕΙΡΙΣΗΣ ΠΕΡΙΕΧΟΜΕΝΟΥ (CMS)

ΕΙΣΑΓΩΓΗ

Σε αυτό το κεφάλαιο αναφέρουμε τα συστήματα διαχείρισης περιεχομένου (CMS). Θα αναλύσουμε τα ανοιχτού και κλειστού κώδικα συστήματα, τα πλεονεκτήματα και ποια τα μειονεκτήματα τους καθώς και τα εργαλεία του καθένα.

Τέλος θα αναφερθούμε και στο σύστημα διαχείρισης περιεχομένου ιστοσελίδων (WCMS), οπού αφορά και την συγκεκριμένη διπλωματική, στους τύπους και τις δυνατότητες του.

ΥΠΟΚΕΦΑΛΑΙΟ 2.1 – ΤΙ ΕΙΝΑΙ ΤΟ CMS

Το CMS (σύστημα διαχείρισης περιεχομένου) είναι μια εφαρμογή λογισμικού που χρησιμεύει στην παρουσίαση και την οργάνωση του περιεχομένου σε ένα πληροφοριακό σύστημα και χρησιμοποιείται για την διαχείριση της ροής εργασίας της ιστοσελίδας.

Επιτρέπει τις αλλαγές του περιεχομένου από τον χρήστη (administrator) χωρίς να είναι απαραίτητες εξειδικευμένες γνώσεις πάνω στην δημιουργία ιστοσελίδων, διότι τα κείμενα γράφονται μέσω online html editors (κειμενογράφους) που επιτρέπουν την μορφοποίηση και την εισαγωγή/εξαγωγή κειμένων όποτε αυτό χρειάζεται.

1. Είδη CMS

Τα συστήματα διαχείρισης περιεχομένου χωρίζονται σε κατηγορίες με βάση το είδος του παρόχου σε Commercial, Open Source και Managed Open Source και με βάση το χώρο αποθήκευσης σε ASP και Licensed.

α. Με βάση το είδος του παρόχου

● Commercial: Οι πάροχοι αυτών των λογισμικών είναι κερδοσκοπικές και μη εταιρίες οι οποίες δημιουργούν αυτό το λογισμικό το οποίο πουλάνε και το υποστηρίζουν τεχνικά.

● Open Source: Δημιουργείται, από έναν συνεργάτη μιας κοινότητας χρηστών, προσφέροντας λύση ενός συστήματος διαχείρισης περιεχομένου. Μετά την δημιουργία, το λογισμικό μοιράζεται στα υπόλοιπα μέλη της κοινότητας για κάποιο συγκεκριμένο σκοπό. Για να συντηρηθεί τεχνικά και να ανανεώνονται οι λειτουργίες του λογισμικού απαιτείται τεχνικό προσωπικό καθώς επίσης το εσωτερικό hardware και το λογισμικό. Η συντήρηση αυτή σαφώς κοστίζει και συμπεριλαμβάνεται στα έξοδα τεχνικής υποστήριξης κάτι που κάνει αυτό το μοντέλο δαπανηρό.

● Manage Open Source: Ένας πάροχος υιοθετεί μια open-source λύση θέτοντας την σαν βασική του πλατφόρμα. Είναι ένας συνδυασμός εμπορικής και ελεύθερης προσέγγισης με συμπληρωματικές υπηρεσίες τεχνικής υποστήριξης. Αυτή η λύση δεν αποτελεί επιλογή σήμερα για τους μη κερδοσκοπικούς παρόχους παρόλο που αρχίζει να εξελίσσεται και οι ειδικοί προβλέπουν πως θα κάνουν πιο έντονη την παρουσία τους.

β. Με βάση το χώρο αποθήκευσης και διαχείρισης

● Licensed: Στα Licensed CMS, δηλαδή στα συστήματα διαχείρισης περιεχομένου με παροχή άδειας, ο πάροχος δεν εγκαθιστά το προϊόν ούτε το ρυθμίζει και το συντηρεί. Αντίθετα, αναλαμβάνει μόνο την πώληση του προϊόντος, παρέχοντας την άδεια χρήσης του και δεν είναι υπεύθυνος για οτιδήποτε άλλο. Είναι εύκολα συντηρήσιμο με μικρό κόστος, από οργανισμούς που ήδη έχουν παρόμοιο είδος υπηρεσίας όπως ένα σύστημα διαχείρισης εξυπηρέτησης πελατών (CRM).

● ASP: Τα Application Service Provider, δηλαδή Υποστήριξης Παρόχου Υπηρεσίας συστήματα, είναι μια λύση η οποία εξαλείφει τα έξοδα αγοράς λογισμικού και hardware καθώς ο κατασκευαστής είναι αυτός που φιλοξενεί τα δεδομένα και το

λογισμικό, σε δικό του server, μειώνοντας τους τεχνικούς πόρους. Το βασικό όμως πλεονέκτημα αυτού του είδους συστήματος είναι η εξέλιξη που προσφέρει ο πάροχος δίνοντας συνεχώς νέες πρωτοποριακές λειτουργίες για την συντήρηση και ανάπτυξη του συστήματος.

2. Πλεονεκτήματα CMS

Τα πλεονεκτήματα των συστημάτων διαχείρισης περιεχομένου ποικίλουν σε χαρακτήρα, υλικό και άυλο.

Συγκεκριμένα, είναι η δυνατότητα, που παρέχουν τα CMS σε πολλά άτομα να έχουν πρόσβαση στη διαχείριση της ιστοσελίδας τους ώστε να την ενημερώνουν συνεχώς με καινούργιο υλικό. Επίσης, ο κάθε υπεύθυνος μπορεί να εστιάσει στο κομμάτι της ιστοσελίδας που ειδικεύεται με όφελος τον καταμερισμό της εργασίας για την προσφορά των μέγιστων δυνατών αποτελεσμάτων.

Στα πλεονεκτήματα θα μπορούσε να προστεθεί το γεγονός πως για να δημιουργήσει κάποιος έναν εντυπωσιακό ιστοχώρο με την χρήση CMS. Οι ανάγκες εκπαίδευσης που απαιτούνται είναι αμυδρές, καθώς μέσω των CMS προσφέρονται έτοιμες φόρμες εισαγωγής/εξαγωγής και μορφοποίησης χωρίς να χρειάζεται κάποιος να έχει εξειδικευμένες γνώσεις για να τις χρησιμοποιήσει. Αυτό έχει σαν αποτέλεσμα να αποφέρει κέρδος στην επιχείρηση ή στον οργανισμό που τα χρησιμοποιεί. Οι τεχνικές γνώσεις είναι μειωμένες επομένως μειώνονται και τα έξοδα για την εκπαίδευση των μελών και αυξάνονται τα οφέλη της επιχείρησης.

Ένα ακόμα πλεονέκτημα είναι πως χρησιμοποιώντας CMS παρατηρείται αύξηση στην επισκεψιμότητα της ιστοσελίδας το οποίο οφείλεται στην άμεση και ανά τακτά χρονικά διαστήματα παροχή και ανανέωση των πληροφοριών που βρίσκονται στην ιστοσελίδα. Αυτό φαίνεται από μία ακόμα δυνατότητα που έχουν τα CMS να παρέχουν στατιστικά στοιχεία επισκεψιμότητας.

Τέλος, βασικό πλεονέκτημα αποτελεί η ευχέρεια δημοσιεύσεων της πληροφορίας σε διάφορα κανάλια. Δηλαδή με αυτοματοποιημένο τρόπο ο χρήστης μπορεί να δημοσιεύσει το περιεχόμενο σε διάφορα τμήματα του διαδικτυακού τόπου του ή σε πολλά σημεία ενός τμήματος, αλλά ακόμα και να δημοσιευτεί το περιεχόμενο σε διάφορες συνεργαζόμενες ιστοσελίδες άλλων οργανισμών.

ΥΠΟΚΕΦΑΛΑΙΟ 2.2 – ΑΝΟΙΧΤΟΥ ΚΩΔΙΚΑ ΣΥΣΤΗΜΑΤΑ ΔΙΑΧΕΙΡΙΣΗΣ ΠΕΡΙΕΧΟΜΕΝΟΥ

Με τον όρο λογισμικό ανοιχτού κώδικα στον τομέα της πληροφορικής, εννοείται ένα λογισμικό στο οποίο ο πηγαίος κώδικας μπορεί να χρησιμοποιηθεί από κάποιον τρίτο. Ανά περιόδους έχουν εμφανιστεί διαφορετικές άδειες χρήσης ειδικά σχεδιασμένες να συνοδεύουν λογισμικό ανοιχτού κώδικα, μερικές από τις οποίες δίνουν τη δυνατότητα στους χρήστες ν' αλλάξουν τον κώδικα βελτιώνοντάς τον, διορθώνοντας πιθανόν λάθη του ή να τον χρησιμοποιήσουν σε άλλες εφαρμογές. Με βάση αυτή τη φιλοσοφία δημιουργήθηκε μία κοινότητα προγραμματιστών και χρηστών, οι οποίοι αναπτύσσουν και επιδιορθώνουν τον κώδικα των προγραμμάτων, φέρνοντας έτσι σε κυκλοφορία νέες βελτιωμένες εκδόσεις λογισμικού. Αξιοσημείωτο είναι ότι στην πλειοψηφία τους τα προγράμματα ανοιχτού κώδικα διατίθενται δωρεάν και χαρακτηρίζονται ελεύθερα.

- **Πλεονεκτήματα Ανοιχτού Κώδικα CMS**

1. Δυνατότητα επιπρόσθετης έρευνας και επεξεργασίας των προγραμμάτων.
2. Προσαρμογή και επέκταση των προγραμμάτων με βάση κάποια ανάγκη.
3. Συντελεί ως εκπαιδευτικό εργαλείο για κάθε χρήστη.
4. Ενσωμάτωση διαδικασιών γενικώς αποδεκτών και αποτελεσματικών.
5. Διαφορετικοί προμηθευτές έχουν τη δυνατότητα ανταλλαγής δεδομένων ετερογενών εφαρμογών και συστημάτων πληροφορικής.
6. Ποικιλία σε εφαρμογές και δοκιμασμένες λύσεις.
7. Προσφέρονται εγγυημένη ασφάλεια και αξιοπιστία, αφού το λογισμικό έχει επεξεργαστεί από πολλούς με αποτέλεσμα να αποφεύγονται σφάλματα.
8. Στην περίπτωση που εντοπιστεί πρόβλημα η επίλυση και η διόρθωση γίνεται σε πολύ σύντομο χρονικό διάστημα.
9. Το κόστος με το οποίο επιβαρύνεται ο χρήστης για ένα πρόγραμμα ανοιχτού κώδικα είναι σχεδόν πάντα μηδενικά.
10. Για τη χρήση ανοιχτού κώδικα δεν δεσμεύεται ο χρήστης ή ο οργανισμός με κάποια από τις εταιρίες.

11. Έχουμε μείωση κόστους αγοράς, χρήσης και συντήρησης των πληροφοριακών λόγω του έντονου ανταγωνισμού.
12. Μηδενικό κόστος σε ανανεώσεις και υποστήριξη.

- **Μειονεκτήματα Ανοιχτού Κώδικα CMS**
  1. Η υποστήριξη των προγραμμάτων ανοιχτού κώδικα γίνεται δύσκολη, καθώς υπάρχει δυσκολία στην εύρεση προσωπικού την ανάλογη τεχνογνωσία.
  2. Υπάρχει ελλιπής τεκμηρίωση και έλλειψη καλού εγχειριδίου χρήσης.
  3. Κάποια προγράμματα ανοιχτού λογισμικού πιθανόν να μην είναι συμβατά με κάποια ήδη γνωστά κλειστά πρότυπα αρχείων και με άλλα συστήματα.
  4. 4.Τα αντίστοιχα εμπορικά προγράμματα προσφέρουν κυρίως καλύτερη υποστήριξη, τεκμηρίωση και συνεργασία με ανώτερες δυνατότητες.
  5. 5.Υπάρχει δυσαναλογία ανάμεσα στο κόστος δημιουργίας και το κόστος συντήρησης για τα προγράμματα ανοιχτού κώδικα.
  6. Με την εγκατάσταση ενός λογισμικού ανοιχτού κώδικα, ο κάθε οργανισμός χρειάζεται να έχει ένα εξαιρετικά καταρτισμένο τμήμα που θα αντιμετωπίζει όποιες δυσκολίες προκύπτουν.
  7. Υπάρχει κόστος επανεκπαίδευσης ατόμων.
  8. Έχουμε έλλειψη εξειδικευμένων εφαρμογών.
  9. Διάφοροι προγραμματιστές προσπαθούν να προωθήσουν τη λύση τους ως την καλύτερη, με αποτέλεσμα να προκύπτει ένα μείγμα λύσεων και το λογισμικό να είναι πολύ αργό και δυσνόητο για τους άλλους προγραμματιστές.
  10. Δεν έχουμε πάντα καινοτομία με τα προγράμματα ανοιχτού κώδικα, αφού πολλές φορές έχουμε την υιοθέτηση της λειτουργικότητας κάποιου άλλου αντίστοιχου.

- **Δημοφιλή Εργαλεία Ανοιχτού Κώδικα CMS**
  1. Joomla: σύστημα διαχείρισης και δημοσίευσης περιεχομένου στο διαδίκτυο.
  2. Drupal: χρησιμοποιείται για δημιουργία και διαχείριση πολλών και διαφορετικών ιστοτόπων.

3. Magento: χρησιμοποιείται για την κατασκευή ηλεκτρονικών καταστημάτων.

ΥΠΟΚΕΦΑΛΑΙΟ 2.3 – ΚΛΕΙΣΤΟΥ ΚΩΔΙΚΑ ΣΥΣΤΗΜΑΤΑ ΔΙΑΧΕΙΡΙΣΗΣ ΠΕΡΙΕΧΟΜΕΝΟΥ

Τα CMS κλειστού κώδικα είναι κάποια συστήματα διαχείρισης περιεχομένου, όπου δεν μπορούμε να επέμβουμε στον πηγαίο κώδικα και να τον τροποποιήσουμε. Η δημιουργία τέτοιων συστημάτων κλειστού κώδικα κοστίζει περισσότερο τόσο σε χρόνο όσο και σε κόστος.

Πλεονεκτήματα Κλειστού Κώδικα CMS

1. Παρέχουν μεγαλύτερη ασφάλεια αφού δεν μπορεί να τροποποιηθεί ο πηγαίος κώδικας παρά μόνο από τους προγραμματιστές.
2. Καλύτερη εκπαίδευση και τεκμηρίωση.
3. Τα περισσότερα CMS κλειστού τύπου είναι ετοιμοπαράδοτα.
4. Υπάρχει μεγαλύτερη υποστήριξη από το εμπόριο, αλλά σε περιορισμένες υπηρεσίες.

Μειονεκτήματα Κλειστού Κώδικα CMS

1. Μόνο οι προγραμματιστές μπορούν να τροποποιήσουν τον πηγαίο κώδικα.
2. Έχουν μεγαλύτερο κόστος για παραμετροποιήσεις, αφού κάτι τέτοιο μπορεί να πραγματοποιηθεί μόνο από τον προγραμματιστή.
3. Υπάρχει κόστος εγκατάστασης.
4. Δεν γίνεται πολύ συχνή ενημέρωση των CMS κλειστού κώδικα
5. Για να τα αποκτήσει κάποιος κοστίζουν περισσότερο.

Δημοφιλή Εργαλεία Κλειστού Κώδικα CMS

1. IBM Workplace Web Content Management
2. Powerfront CMS
3. Vignette Content Management
4. Jalios JCMS

ΥΠΟΚΕΦΑΛΑΙΟ 2.3 – ΣΥΣΤΗΜΑΤΑ ΔΙΑΧΕΙΡΙΣΗΣ ΠΕΡΙΕΧΟΜΕΝΟΥ ΙΣΤΟΥ

Το WCMS (Web Content Management System) ή αλλιώς σύστημα διαχείρισης περιεχομένου ιστού είναι ουσιαστικά μια εφαρμογή (Web Application) για την δημιουργία και την διαχείριση δυναμικής συλλογής περιεχομένου και υλικού εγγράφων HTML.

Το WCMS όπως και το CMS επιτρέπει στον διαχειριστή να επεξεργαστεί το περιεχόμενο της ιστοσελίδας χωρίς να έχει γνώσης προγραμματισμού. Επίσης παρέχει την δυνατότητα στον χρήστη να αποθηκεύσει το περιεχόμενο και ως XML για την εύκολη παρουσίαση του και την διευκόλυνση στο να ξαναχρησιμοποιηθεί.

Τέλος, η διαχείριση του WCMS γίνεται μέσω ενός φυλλομετρητή (browser) με εξαίρεση κάποιων συστημάτων τα οποία απαιτούν πρόσβαση στον server όπου φιλοξενείται η εφαρμογή.

Δυνατότητες WCMS

Τα WCMS (συστήματα διαχείρισης περιεχομένου ιστού) έχουν την δυνατότητα να επεξεργάζονται και να διαχειρίζονται έγγραφα και χρονοδιαγράμματα του διαδικτύου.

Έχουν λοιπόν τις εξής δυνατότητες:

- Αυτοματοποιημένα πρότυπα:
  Επιτρέπει την δημιουργία τυποποιημένων προτύπων που συνήθως είναι γραμμένα σε HTML γλώσσα και εφαρμόζονται αυτόματα σε ένα ήδη υπάρχον περιεχόμενο του οποίου οι αλλαγές γίνονται από ένα κεντρικό σημείο.

- Έλεγχος πρόσβασης:
  Με αυτή τη λειτουργία ελέγχεται η πρόσβαση των χρηστών στην ιστοσελίδα. Αυτό σημαίνει ότι, μόνο εγγεγραμμένοι χρήστες μπορούν να αλληλοεπιδρούν με την ιστοσελίδα και σημεία αυτής. Συνήθως οι ανώνυμοι χρήστες μπορούν να αλληλοεπιδράσουν με την ιστοσελίδα με περιορισμένο περιεχόμενο που προσφέρεται σε αυτούς

- Κλιμακωτή επέκταση:
  Η κλιμακωτή επέκταση υπάρχει στα πιο σύγχρονα WCMS και είναι διαδικτυακές πύλες μέσα σε μια ιστοσελίδα.

- Επεξεργάσιμο περιεχόμενο:
  Με αυτή την λειτουργία το περιεχόμενο μπορεί να επεξεργαστεί ευκολότερα και πιο γρήγορα σε τοπικό επίπεδο και από μη τεχνικούς χρήστες.

- Κλιμακωτό χαρακτηριστικό σετ:
  Δίνεται η δυνατότητα για την χρήση πρόσθετων και επεκτάσεων που κάνουν καλύτερη την λειτουργικότητα ενός ιστότοπου.

- Αναβαθμίσεις προτύπων του Παγκόσμιου Ιστού:
  Συχνή ενημέρωση με νέα χαρακτηριστικά των προτύπων ενός ιστού.

- Διαχείριση ροής εργασίας:
  Είναι μια διαδικασία κατά την οποία ο δημιουργός ενός περιεχομένου υποβάλει ένα κείμενο του οποίου το αντίγραφο εάν δεν καθαριστεί και δε εγκριθεί δεν δημοσιεύεται. Αυτή η διαδικασία επιτυγχάνεται μέσω των κύκλων εργασιών των CMS.

- Συνεργασία:
  Πολλοί εξουσιοδοτημένοι χρήστες μπορούν να ανακτήσουν και να επεξεργαστούν το περιεχόμενο ακόμα και να σχολιάσουν.

- Αντιπροσωπία:
  Κάποια λογισμικά CMS παρέχουν περιορισμένα δικαιώματα στο περιεχόμενο για ορισμένες ομάδες χρηστών δίνοντας με αυτόν τον τρόπο ευθύνη για την διαχείριση του.

- Η διαχείριση των εγγράφων:
  Τα έγγραφα μέσω των CMS μπορούν να δημοσιευθούν, να αρχειοθετηθούν, να αναθεωρηθούν ακόμα και να καταστραφούν.

- Πολυγλωσσικό:
  Παρέχεται η δυνατότητα εμφάνισης του περιεχομένου σε πολλές γλώσσες.

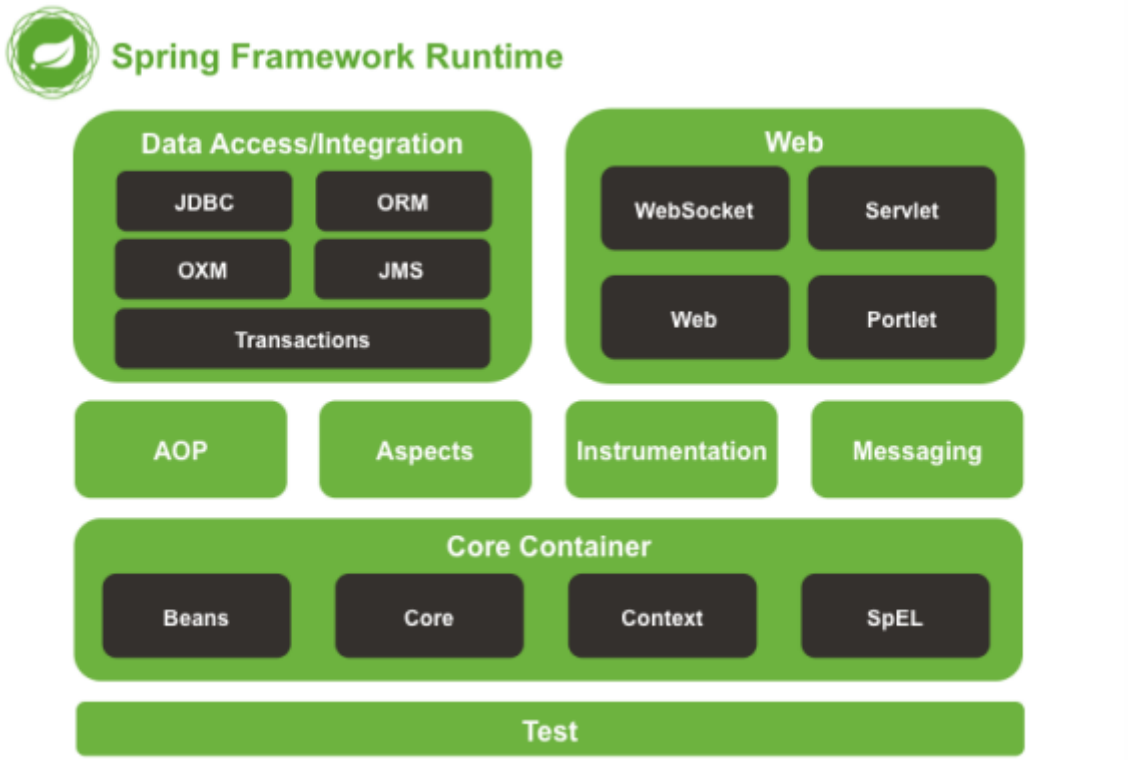
- Εκδόσεις:
  Επιτρέπεται η ανάκτηση προηγούμενων εκδόσεων. Χρήσιμο σε περίπτωση που το περιεχόμενο αλλάζει και απαιτείται ενημέρωση.

ΚΕΦΑΛΑΙΟ 3 – ΤΕΧΝΟΛΟΓΙΕΣ ΚΑΙ ΕΡΓΑΛΕΙΑ ΑΝΑΠΤΥΞΗΣ ΙΣΤΟΤΟΠΟΥ

ΕΙΣΑΓΩΓΗ

Σε αυτό το κεφάλαιο παρουσιάζονται αναλυτικά όλες οι τεχνολογίες που χρησιμοποιήθηκαν για την υλοποίηση της εφαρμογής.

ΥΠΟΚΕΦΑΛΑΙΟ 3.1 –SPRING FRAMEWORK



Εικόνα 1 - Η αρχιτεκτονική του Spring

Spring MVC:

Το Spring είναι μια ανοιχτού κώδικα (open source) πλατφόρμα για εφαρμογές Java. Η πρώτη του κυκλοφορία έγινε το 2003. Αναπτύχθηκε για να αντιμετωπίσει την πολυπλοκότητα ανάπτυξης επαγγελματικών (enterprise) εφαρμογών. Τα κυριότερα

χαρακτηριστικά του μπορούν να χρησιμοποιηθούν σε Java εφαρμογές, αλλά με την δυνατότητα των διαθέσιμων επεκτάσεων του Spring μπορούμε να δημιουργήσουμε και Web εφαρμογές. Με την χρήση μικρών και απλών σε κώδικα Java Beans, το Spring κάνει εφικτή την υλοποίηση πραγμάτων που μέχρι τότε μπορούσαν να γίνουν μόνο μέσω επαγγελματικών Java Beans. Το Spring δεν είναι μόνο χρήσιμο για την ανάπτυξη εφαρμογών στην πλευρά του εξυπηρετητή, αλλά μπορεί να βοηθήσει στην απλοποίηση του κώδικα, τον ευκολότερο και πιο αποτελεσματικό έλεγχο και τη χαλαρή διασύνδεση κάθε Java εφαρμογής.

Το Spring συμπεριλαμβάνει τις εξής μονάδες/ενότητες.

- Inversion of Control Container – Dependency Injection:

  Είναι το κεντρικό κομμάτι του Spring όπου είναι και η αρχή σχεδίασης του κώδικα του Spring. Παρέχει τα μέσα για τις ρυθμίσεις και την διαχείριση των Java αντικειμένων. Ουσιαστικά, είναι υπεύθυνο για την διαχείριση του κύκλου ζωής των αντικειμένων, την δημιουργία και την κλήση των μεθόδων.
  Τα δημιουργημένα αντικείμενα του container λέγονται beans και ρυθμίζονται φορτώνοντας XML αρχεία που περιέχουν ορισμούς για κάθε bean.

- Aspect-Oriented Programming Framework:

  Συμπληρώνει την αντικειμενοστρέφεια στον προγραμματισμό. Η κύρια μονάδα του αντικειμενοστρεφή προγραμματισμού είναι η κλάση ενώ στο AOP είναι η όψη (Aspect). Η χρήση του AOP δεν είναι απαραίτητη στο Spring. Το AOP χρησιμοποιείται για να προσφέρει μια δηλωτική υπηρεσία για το EJB. Γνώστη υπηρεσία EJB είναι αυτή των συναλλαγών.

- Data Access Framework

  Το συγκεκριμένο framework βοηθάει στην επικοινωνία εφαρμογών και βάσης δεδομένων. Τέτοια υποστήριξη παρέχεται στα περισσότερα data access frameworks όπως είναι το JDBC, JPA, OJB, κ.α.
  Το Spring διαθέτει χαρακτηριστικά για κάθε data access framework όπως είναι η διαχείριση πόρων, διαχείριση συναλλαγών, exceptions, κ.α. Τα χαρακτηριστικά αυτά είναι διαθέσιμα όταν κάνουμε χρήση των κλάσεων που προσφέρονται από το Spring. Το κύριο πλεονέκτημα αυτών των εργαλείων

είναι ότι μπορείς να χρησιμοποιείς APIs όπως το Hibernate για απευθείας επικοινωνία με την βάση.

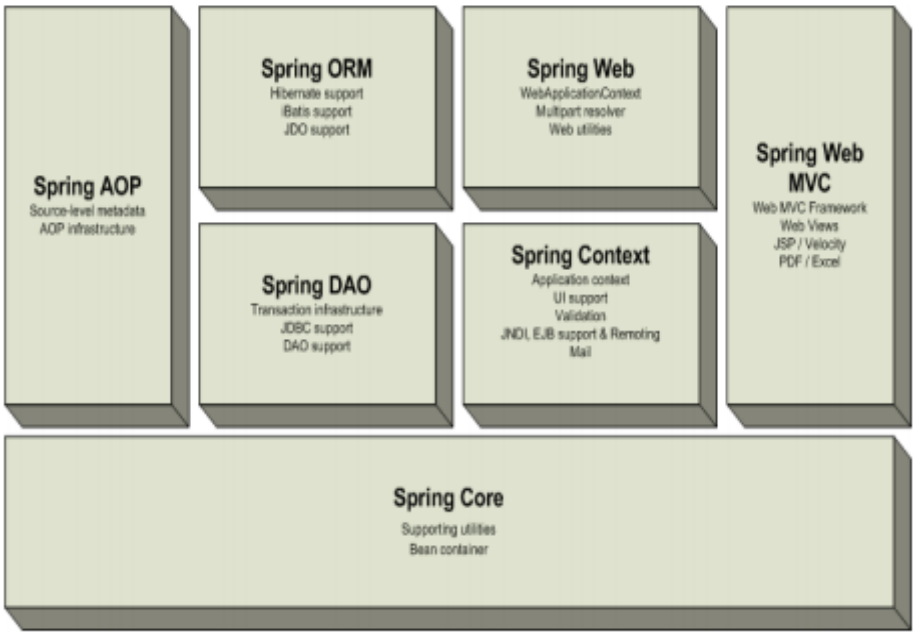
- Transaction Management Framework:
  Είναι το framework το οποίο επιτρέπει συναλλαγές μεταξύ δύο συστημάτων σε τοπικό και διαδικτυακό επίπεδο. Το Spring παρέχει έναν Platform Transaction Manager που μπορεί να χρησιμοποιηθεί για έναν μεγάλο αριθμό από στρατηγικές για transaction management όπως οι συναλλαγές που γίνονται με JDBC. Με το Spring Data Access και το Transaction Management framework - το Data access υλοποιεί ουσιαστικά το transaction management framework.

- MVC - Model View Controller Framework:
  Το MVC είναι ένα αρχιτεκτονικό πρότυπο λογισμικού για την ανάπτυξη διεπαφών χρήστη στους υπολογιστές. Το MVC είναι σχεδιασμένο έτσι ώστε να επιτρέπει την επαναχρησιμοποίηση του κώδικα. Το MVC βασίζεται στην δημιουργία αιτημάτων. Ορίζει interface στρατηγικής για κάθε ευθύνη που πρέπει να αναλάβει ένα framework που βασίζεται σε αιτήματα (requests). Σκοπός του είναι να επιτρέπει σε προγραμματιστές να κάνουν την δικιά του υλοποίηση στα αιτήματα, αν αυτό κριθεί απαραίτητο για διεκπεραίωση της ανάπτυξης μιας εφαρμογής. Όλα τα interfaces είναι άρρηκτα συνδεδεμένα με το Servlet API.

Τα σημαντικότερα interfaces του Spring είναι τα παρακάτω:

1. Controller: Ο controller είναι κομμάτι που βρίσκεται ενδιάμεσα από το model και το view και είναι υπεύθυνο για να διαχειρίζεται τα εισερχόμενα αιτήματα και να ανακατευθύνει στην κατάλληλη απάντηση.
2. HandlerAdapter: Υπεύθυνο για την διαχείριση των εισερχομένων αιτημάτων
3. HandlerInterceptor: Υπεύθυνο για το φιλτράρισμα των αιτημάτων.
4. HandlerMapping: Υπεύθυνο για την επιλογή αντικειμένων που θα διαχειριστούν τα αιτήματα.
5. LocaleResolver: Αποθηκεύει το locale ενός χρήστη
6. MultipartResolver: Διευκολύνει την διαδικασία του να δουλεύεις με αρχεία
7. View: Υπεύθυνο για απαντήσεις αιτημάτων.

Όλα τα παραπάνω αποτελούν το framework. Οι δυνατότητες που προσφέρει το Spring είναι ισχυρές και προσφέρουν μεγάλες ευκολίες στην ανάπτυξη μιας εφαρμογής. Οι ενότητες που ήδη αναφέραμε, σαν σύνολο δίνουν στον προγραμματιστή ότι χρειάζεται, για να αναπτύξει επαγγελματικές εφαρμογές, παρέχοντας σε αυτούς την δυνατότητα να επιλέξουν όποιες από τις ενότητες πιστεύουν ότι καλύπτουν τις ανάγκες τους. Επιπλέον παρέχονται τρόποι σύνδεσης με άλλα πλαίσια και βιβλιοθήκες, έτσι ώστε ο προγραμματιστής να μην χρειαστεί να το αναπτύξει από την αρχή.



Εικόνα 2- Ενότητες του Spring

Το Spring MVC είναι βασισμένο στην αρχιτεκτονική model-view-controller. Όπου :

- Μοντέλο: Στο μοντέλο τοποθετούμε τις λειτουργίες της εφαρμογής που σχετίζονται με την πρόσβαση στην βάση δεδομένων.
- Όψη: Μέσα στην όψη υπάρχει η HTML της σελίδας της εφαρμογής μας. Ουσιαστικά η όψη είναι αυτό που βλέπει ένας χρήστης.

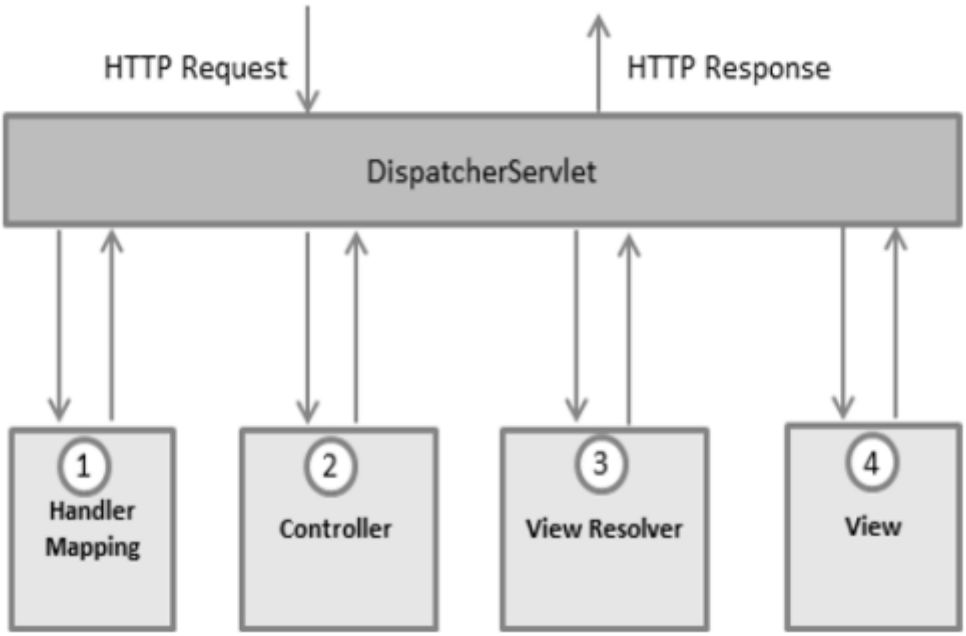
- Ρυθμιστής: Ο ρυθμιστής είναι ο μεσίτης μεταξύ του μοντέλου και της όψης. Ελέγχει πως «τρέχει» η εφαρμογή, μιλάει με το μοντέλο, παίρνει τα δεδομένα που ζητά και εν συνεχεία αφού τα επεξεργαστεί τα στέλνει πίσω στην όψη για απεικόνιση.

Πλεονεκτήματα Spring:

- Επεκτασιμότητα: Μπορούμε μελλοντικά να προσθέσουμε λειτουργίες ή να αλλάξουμε κάποιες από τις ήδη υπάρχουσες λειτουργίες και να έχουμε άλλα αποτελέσματα. Τα προγράμματα που είναι φτιαγμένα με MVC αρχιτεκτονική έχουν βασικό χαρακτηριστικό ότι είναι επεκτάσιμα.
- Διαχωρισμός Προβλημάτων: Το κάθε επίπεδο επιτελεί ξεχωριστό έργο και ταυτόχρονα συνεργάζεται με τα άλλα επίπεδα. Μία σωστή MVC εφαρμογή είναι εκείνη που τα τρία επίπεδα είναι ξεκάθαρα καθορισμένα και δεν συμπλέκονται.
- Ελεγξιμότητα: Οι MVC εφαρμογές έχουν την δυνατότητα να είναι ελέγξιμες και με τον τρόπο αυτό συντηρούνται πιο εύκολα. Ας κάνουμε ένα απλό παράδειγμα. Έστω ότι έχουμε μία εφαρμογή η οποία διαθέτει μία λειτουργία σύνδεσης (login), δηλαδή ζητά από τον χρήστη να πληκτρολογήσει κάποια στοιχεία σε μία φόρμα και εν συνέχεια τα εισάγει μέσα στο σύστημα. Αυτή τη λειτουργία την ελέγχει κάποιος ρυθμιστής ο οποίος περιέχει κώδικα που διαχειρίζεται τα δεδομένα αυτά που εισήχθησαν από τον χρήστη. Αυτός ο ρυθμιστής θεωρείται μία «μονάδα» (unit). Στα MVC πλαίσια μπορούμε με πολλή ευκολία να γράψουμε απλό κώδικα ελέγχου με τον οποίο τεστάρουμε αυτόν τον ρυθμιστή αλλά και κάθε μία από τις λειτουργίες του. Στην συνέχεια παίρνουμε τα αποτελέσματα και βλέπουμε η συγκεκριμένη μονάδα της εφαρμογής μας αν λειτουργεί σωστά.
- URLs: Τα περισσότερα MVC πλαίσια για εφαρμογές του διαδικτύου (web applications) δίνουν τη δυνατότητα να έχουμε «καθαρά» URLs. Με το MVC μπορούμε να έχουμε πιο όμορφα και εύληπτα URLs από το χρήστη αλλά και από την μηχανή αναζήτησης.
- Tools: Ένα άλλο πλεονέκτημα είναι ότι το Spring περιέχει στον πυρήνα του υπάρχουσες τεχνολογίες όπως είναι το ORM framework, logging framework

και άλλα πολλά τα οποία είναι απαραίτητα για την δημιουργία εφαρμογών οπότε δεν είναι απαραίτητη χειροκίνητη ένταξη τους στο project που δουλεύουμε κάθε φορά.

Το Spring MVC είναι σχεδιασμένο πάνω σε ένα Dispatcher Servlet ο οποίος διαχειρίζεται όλες τις HTTP αιτήσεις (requests) και απαντήσεις (responses). Η επεξεργασία των αιτήσεων αναπαριστάται στην παρακάτω εικόνα.



Εικόνα 3 - Επεξεργασία αιτήσεων από τον Dispatcher Servlet

Από την στιγμή που ο Dispatcher Servlet θα λάβει μία HTTP αίτηση θα ακολουθήσουν οι παρακάτω ενέργειες.

1. Χειριστής Απεικόνισης (Handler Mapping): Θα κληθεί ο κατάλληλος ρυθμιστής για να διαχειριστεί την αίτηση.

2. Ρυθμιστής: Θα λάβει την αίτηση και θα καλέσει την κατάλληλη μέθοδο – υπηρεσία ανάλογα με το είδος της αίτησης (POST ή GET). Η μέθοδος θα δημιουργήσει τα δεδομένα του μοντέλου (model data) βασισμένα στην επιχειρησιακή λογική της εφαρμογής (business logic) και θα επιστρέψει το όνομα της όψης στον Dispatcher Servlet.

3. Μετατροπέας Όψης (View Resolver): Θα βοηθήσει τον Dispatcher να χρησιμοποιήσει την κατάλληλη όψη.

4. Όψη: Ο Dispatcher θα περάσει τα δεδομένα του μοντέλου στην όψη, η οποία τελικώς θα εμφανιστεί στο πρόγραμμα περιήγησης του χρήστη.

Spring Boot:

Το Spring Boot είναι μια πλήρως εξοπλισμένη σουίτα με την οποία ένας προγραμματιστής μπορεί εύκολα να χτίσει ένα project γρήγορα χωρίς να χρειάζεται να ρυθμίσει και να κατασκευάση χειροκίνητα την δομή του project. Έτσι, με την βοήθεια αυτού, κατασκευάζουμε γρήγορα μια εφαρμογή Spring MVC είτε αυτή είναι Web είτε desktop, γράφοντας όσο το δυνατόν λιγότερες γραμμές κώδικα για την αρχικοποίηση της εφαρμογής.

Hibernate:

Το Hibernate αποτελεί μία βιβλιοθήκη ελεύθερου λογισμικού για τη Java. είναι μια ολοκληρωμένη λύση στο πρόβλημα της διαχείρισης δεδομένων, μμεσολαβώντας στις αλληλεπιδράσεις της εφαρμογής με την σχεσιακή βάση δεδομένων, επιτρέποντας στον προγραμματιστή να ασχοληθεί με μόνο με τις λειτουργίες της. Ουσιαστικά, είναι μια υλοποίηση της διεπαφής που ορίζεται στα πλαίσια του Java Persistence API (JPA). Η τεχνολογία JPA ορίζει ένα πλαίσιο για την αντιστοίχιση οντοκεντρικών μοντέλων σε σχήματα παραδοσιακών σχεσιακών βάσεων δεδομένων. Κεντρικός στόχος για την ανάπτυξή της υπήρξε ο ορισμός ενός ενιαίου τρόπου για την διατήρηση δεδομένων σε όλες τις Java εφαρμογές. Το Hibernate συγκαταλέγεται στην κατηγορία του λογισμικού Object-Relational Mapping (ORM).

Το λογισμικό ORM προσφέρει την χρησιμοποίηση μιας σχεσιακής βάσης δεδομένων σαν να ήταν αντικειμενοστραφής. Έτσι ο προγραμματιστής χρησιμοποιεί τα αντικείμενα της συγκεκριμένης εφαρμογής, τα τροποποιεί σχετικά με τη λογική της εφαρμογής που αναπτύσσει και τα τροποποιεί στην βάση ως αντικείμενα. Σε αυτό το σημείο είναι το Hibernate που γνωρίζοντας την αντιστοιχία μεταξύ βάσης και λογικής της εφαρμογής, αναλαμβάνει να κατασκευάσει την κατάλληλη εντολή (query) η οποία και στέλνεται στην βάση δεδομένων. Έπειτα τα

αποτελέσματα που επιστρέφει η βάση, το Hibernate τα επιστρέφει στον προγραμματιστή ως αντικείμενα της εφαρμογής. Είναι δηλαδή ένα ενδιάμεσο επίπεδο μεταξύ της εφαρμογής και της βάσης δεδομένων.

Το Hibernate είναι μία ολοκληρωμένη λύση στο πρόβλημα της διαχείρισης δεδομένων. Είναι αυτό το οποίο εκτελεί τις ενέργειες μεταξύ εφαρμογής-βάσης εκτελώντας όλες τις ενέργειες μεταξύ των εφαρμογών και των σχεσιακών βάσεων και έτσι αποτρέπει τον κίνδυνο αστοχιών στην αποτύπωση σχέσεων μεταξύ κλάσεων και βάσης.

Πλεονεκτήματα Hibernate:

- Παραγωγικότητα: Στην ανάπτυξη λογισμικού ένα μεγάλο μέρος της προγραμματιστικής προσπάθειας αφιερώνεται στην διεπαφή της εφαρμογής με την βάση δεδομένων. Το Hibernate αυτοματοποιώντας τις βασικές λειτουργίες Create-Read-Update-Delete (CRUD) επιτρέπει αρχικά στον προγραμματιστή να επικεντρώνει τη προσπάθεια του στη λογική της εφαρμογής. Επίσης, υπάρχει η δυνατότητα να ακολουθήσουν δύο στρατηγικές ανάπτυξης λογισμικού : είτε αρχίζοντας από το μοντέλο δεδομένων είτε από την βάση δεδομένων. Αυτό μειώνει σε μεγάλο βαθμό το χρόνο ανάπτυξης.

- Συντηρησιμότητα: Με την χρήση του Hibernate γράφονται σημαντικά λιγότερες γραμμές κώδικα και ο κώδικας είναι πιο κατανοητός και καλογραμμένος. Αυτό κάνει την συντήρηση της εφαρμογής ευκολότερη.

- Ανεξαρτησία από την βάση δεδομένων: Με την συμβατότητα του Hibernate με διαφορετικές βάσεις δεδομένων και την δυνατότητα σύνδεσης του με την βάση μέσω δηλώσεων ορισμένων σε ειδικό αρχείο, η αναπτυσσόμενη εφαρμογή μπορεί με ελάχιστες τροποποιήσεις να χρησιμοποιηθεί με βάσεις δεδομένων διαφορετικών κατασκευαστών

ΥΠΟΚΕΦΑΛΑΙΟ 3.2 – JAVASCRIPT/ANGULAR

Η **JavaScript** είναι μια γλώσσα συγγραφής σεναρίων (scripting language) που χρησιμοποιείται για να προσθέσει εφέ, διαλογικότητα, αλληλεπίδραση, διαδραστικότητα (interactivity) στις ιστοσελίδες μας και είναι ανταγωνιστική της γλώσσας προγραμματισμού VBScript. Δημιουργήθηκε από την εταιρία Netscape και το αρχικό της όνομα ήταν LiveScript. Ο κώδικας της JavaScript γράφεται σε καθαρό κείμενο (ASCII μορφή) και ενσωματώνεται μέσα στον κώδικα της HTML, μπορεί δε να εκτελεστεί αμέσως ή όταν λαμβάνει χώρα ένα συμβάν (event). Δεν γίνεται μεταγλώττιση (compilation) του κώδικα της JavaScript, αρκεί μόνο το πρόγραμμα περιήγησης να υποστηρίζει την JavaScript. Η γλώσσα υποστηρίζει αντικειμενοστραφή προγραμματισμό και χρήση συναρτήσεων και η εκτέλεση της γίνεται στην πλευρά του πελάτη (client-side), δηλαδή στο πρόγραμμα περιήγησης του χρήστη και με αυτόν τον τρόπο δεν επηρεάζει την απόδοση του εξυπηρετητή. Αν και ακούγονται ίδιες, η Java και η JavaScript δεν έχουν καμία απολύτως σχέση μεταξύ τους, ούτε στη σύνταξή τους σαν γλώσσες προγραμματισμού ούτε και στις εφαρμογές που χρησιμοποιούνται.

Σύγχρονα JavaScript πλαίσια λογισμικού για την ανάπτυξη και τη συντήρηση κώδικα (frameworks) όπως τα Angular JS, React JS, κ.α. δίνουν δυνατότητες για δημιουργία διαφόρων εφαρμογών Διαδικτύου. Η εξέλιξη της JavaScript έδωσε εξαιρετικά εργαλεία και μεθοδολογίες για την δημιουργία web εφαρμογών και αποτελεί πλέον μονόδρομο στην υλοποίηση εφαρμογών σε περιβάλλον περιηγητή.

Η **Angular**, όπως αναφέρεται από τη 2η έκδοση και μετά, μπορεί να αναπτυχθεί χρησιμοποιώντας JavaScript, Dart και TypeScript, όμως οι περισσότερες πηγές και παραδείγματα που υπάρχουν στο Διαδίκτυο χρησιμοποιούν την TypeScript για την ανάπτυξη εφαρμογών, η οποία εν συντομία είναι ένα υπερσύνολο της JavaScript και χρησιμοποιεί έτοιμες και σωστές πρακτικές αποφεύγοντας λάθη, επαναλήψεις και δυσανάγνωστο κώδικα που συχνά βλέπουμε σε μεγάλες εφαρμογές, καθιστώντας πιο δύσκολη τη συντήρησή τους. Η Angular αποτελεί ένα ολοκληρωμένο framework που προσφέρει δυνατότητες δρομολόγησης (routing) κάτι που λείπει από τα υπόλοιπα JavaScript frameworks τα οποία καλύπτουν αυτή την ανάγκη με εξωτερικά εργαλεία. Επίσης, η Angular είναι πολύ αποδοτική στην

κατανάλωση μικρουπηρεσιών (microservices), μίας αρχιτεκτονικής που αφορά μικρές σε μέγεθος, ευέλικτες και ανεξάρτητες μεταξύ τους με σκοπό τη Συνεχή Παράδοση (Continuous Delivery) λογισμικού.

Μια εφαρμογή μίας σελίδας (ΕΜΣ) «τραβάει» μόνο μία HTML σελίδα από τον εξυπηρετητή (server), όταν ο χρήστης αρχίζει να χρησιμοποιεί την εφαρμογή/σελίδα. Μαζί με αυτή τη σελίδα, ο εξυπηρετητής στέλνει και μια «μηχανή» για την εφαρμογή στον πελάτη (browser). Η μηχανή αυτή ελέγχει την επεξεργασία, τα εισερχόμενα κι εξερχόμενα δεδομένα μέσω μικρο-υπηρεσιών, τη διαμόρφωση και τη φόρτωση των σελίδων της εφαρμογής. Αυτό έχει σαν αποτέλεσμα η πλειοψηφία του κώδικα να τρέχει στον περιηγητή και ο σημαντικά λιγότερος υπόλοιπος στον εξυπηρετητή. Επειδή πλέον μειώνεται η εξάρτηση από τον εξυπηρετητή, όσοι χρήστες κι αν μπουν στην εφαρμογή, η απόδοση δεν επηρεάζεται.

Η Angular μπορεί να αναπτυχθεί κυρίως χρησιμοποιώντας (αλλά όχι αποκλειστικά) **TypeScript**, το οποίο είναι ένα υπερσύνολο της JavaScript. Η TypeScript είναι η γλώσσα στην οποία δημιουργήθηκε το πλαίσιο λογισμικού Angular. Η TypeScript είναι μία γλώσσα ανοιχτού κώδικα που δημιουργήθηκε και συντηρείται από τη Microsoft. Πρόκειται ουσιαστικά για ένα υπερσύνολο της JavaScript (Ecmascript 5 - ES5) με αυστηρή και στατική σύνταξη το οποίο μεταγλωττίζεται σε JavaScript κώδικα. Παράγει πιο οργανωμένο κώδικα JavaScript και είναι εύκολο να ξεκινήσει κανείς να δουλεύει σε Angular μέσω αυτής.

Υπάρχουν πέντε βασικές έννοιες στην Angular:

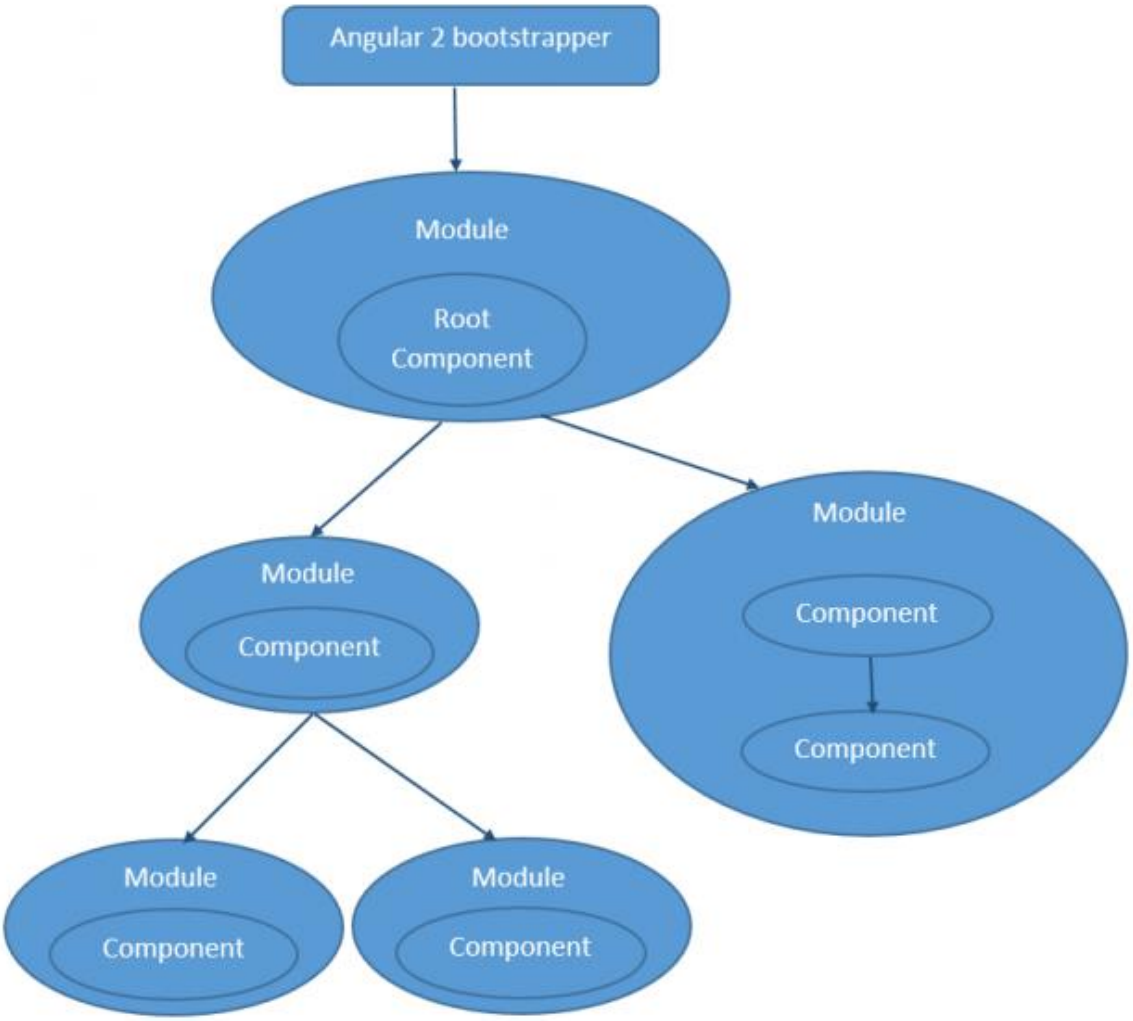
- Λειτουργικές Μονάδες – Modules:
  Μηχανισμός που ομαδοποιεί τεχνητά HTML στοιχεία και υπηρεσίες ώστε να χρησιμοποιηθούν για την ολοκλήρωση της εφαρμογής
- Συστατικά – Components:
  Αποτελείται από την κλάση και το πρότυπο
- Πρότυπα – Templates:
  Το πρότυπο είναι ένα κομμάτι HTML κώδικα με συντακτικές προσθήκες της Angular

- Υπηρεσίες – Services:

  Είναι τυπικά μία TypeScript κλάση η οποία δίνει την δυνατότητα να καταναλώσουμε δεδομένα μέσω http κλήσεων των REST υπηρεσιών.

- Δρομολόγηση – Routing:

  Η δρομολόγηση σε μια ιστοσελίδα διαδικτύου αφορά την πλοήγηση σε διαφορετικές όψεις της ιστοσελίδας.



Εικόνα 4 - Δομή εφαρμογής Angular

ΥΠΟΚΕΦΑΛΑΙΟ 3.3 – HTML

Η γλώσσα σήμανσης υπερκειμένου **HTML** (Hyper Text MarkUp Language) είναι μια γλώσσα η οποία «λέει» στον υπολογιστή πως θα πρέπει να εμφανίσει μια ιστοσελίδα. Είναι η πιο απλοϊκή μορφή που θα μπορούσε να πάρει μια ιστοσελίδα. Ξεκίνησε σχεδόν άμεσα η ιστορία της με την ανάπτυξη και τη σύνδεση των υπολογιστών. Χρησιμοποιήθηκε αρχικά για την παρουσίαση απλών ηλεκτρονικών εγγράφων , για αυτό και η πρωταρχική ονομασία web pages. Σήμερα , βέβαια, η HTML έχει φτάσει στην version 5.0 ικανή και για να υποστηρίξει browsing μέσω έξυπνων τηλεφώνων, τάμπλετ και άλλων συσκευών αφής και όχι μόνο. Αλλά παρόλα αυτά, χωρίς τα θεμέλια που είναι η HTML , δεν θα μπορούσε η εκάστοτε ιστοσελίδα να έχει την μορφή που σήμερα απολαμβάνουμε ως χρήστες και επισκέπτες. Ας δούμε κάποια στοιχεία της για να καταλάβουμε τι εστί HTML. Η HTML , λοιπόν, περιλαμβάνει στοιχεία τα οποία ονομάζονται ετικέτες(tags) και διακρίνονται σε tags αρχής και tag τέλους . Με το όνομα_ετικέτας δηλώνονται στο browser η ενέργεια που θα εκτελεστεί στο κείμενο.

Παρακάτω δίνεται ένα απλό HTML παράδειγμα:

```
<HTML>
<HEAD>
<TITLE> ΠΑΡΑΔΕΙΓΜΑ </TITLE>
<!-- Δεκέμβριος 1998. Ι. Η. Λαγαρής  -->
</HEAD>
<BODY>
<H1> ΠΑΡΑΔΕΙΓΜΑ </H1>
Τα αρχεία σε HTML είναι απλά αρχεία και μπορούν να
κατασκευαστούν από οποιονδήποτε συντάκτη κειμένου. <P>
Τέλος.
</BODY>
</HTML>
```

Εικόνα 5 - Παράδειγμα κώδικα HTML

Και η επεξήγηση των παραπάνω ετικετών είναι η εξής:

- <HTML>: η ετικέτα αυτή δηλώνει ότι το αρχείο περιέχει κώδικα σε HTML.

- <HEAD>: προσδιορίζει το περιεχόμενο της ιστοσελίδας όπως ο τίτλος της, η γλώσσα που προσδιορίζει η σελίδα, διάφορα keywords κ.ά.
- <TITLE> : περικλείεται ο τίτλος της ιστοσελίδας.
- </TITLE>: κλείνει το tag title.
- </HEAD> : κλείνει το tag head.
- <BODY> : γράφεται ο κώδικας που αφορά την εμφάνιση της ιστοσελίδας μας στο browser.
- <H> : ορίζει μια επικεφαλίδα.
- </H> : κλείνει το tag h.
- : ορίζει μια νέα παράγραφο.
- : κλείνει το tag p.
- </BODY> : κλείνει το tag body.
- </HTML> : κλείνει το tag html.

Αξίζει να αναφερθεί ότι υπάρχουν και οι ετικέτες οι οποίες χρησιμοποιούνται για να μορφοποιούν αναλόγως τους διάφορους τίτλους κειμένων και παραγράφων μέσα στην ιστοσελίδα μας. Η χρήση αυτών, είναι καθαρά στη δικαιοδοσία του κατασκευαστή και επιτυγχάνει την καλύτερη ανάγνωση των άρθρων αλλά και την εστίαση της προσοχής του χρήστη/επισκέπτη σε συγκεκριμένα σημεία.

ΥΠΟΚΕΦΑΛΑΙΟ 3.4 – CSS/BOOTSTRAP

Η **CSS** (Cascading Style Sheets) (Διαδοχικά Φύλλα Στυλ) είναι μια γλώσσα υπολογιστή που ανήκει στην κατηγορία των γλωσσών φύλλων στυλ που χρησιμοποιείται για τον έλεγχο της εμφάνισης ενός εγγράφου που έχει τις γλώσσες HTML και XHTML. Ελέγχει δηλαδή την εμφάνιση μιας ιστοσελίδας και γενικότερα ενός ιστότοπου. Κάποια πλεονεκτήματα της χρήσης CSS είναι ότι έχει πιο εύκολα διαχειρίσιμο κώδικα και κάνει την πλοήγηση γρηγορότερη. Το Φύλλο Στυλ που εφαρμόζεται σ' ένα έγγραφο μπορεί να προέρχεται από: • το συγγραφέα μιας ιστοσελίδας • το χρήστη του πλοηγού • τον ίδιο τον πλοηγό 36 Εικόνα 12 - Παράδειγμα CSS {Πηγή: https://agendamahala.com/introduction-to-css } Στην παραπάνω εικόνα αυτό που ουσιαστικά εκτελείται είναι η μορφοποίηση των επικεφαλίδων h1 της HTMl καθώς και των παραγράφων μέσα στην ετικέτα

της HTML. Συγκεκριμένα, ορίζεται συγκεκριμένος τύπος γραμματοσειράς (font-family), μέγεθος και χρώμα γραμματοσειράς (font-size, color). Βέβαια, εννοείται ότι υπάρχουν και άλλες μορφοποιήσεις που μπορούμε να επιτύχουμε σε κείμενο και όχι μόνο όπως σκιά, περίγραμμα, φόντο κοκ.

Το **Bootstrap** είναι ένα καλαίσθητο, ισχυρό μετωπιαίο (front) περιβάλλον για ταχύτερη και ευκολότερη ανάπτυξη ιστοσελίδων. Το Bootstrap βασίζεται στις HTML, CSS και JavaScript και υποστηρίζει όλα τα γνωστά προγράμματα περιήγησης και όλες σχεδόν τις εκδόσεις τους. Το Bootstrap αναπτύχθηκε από τους Mark Otto και Jacob Thornton για λογαριασμό του Twitter ως ένα περιβάλλον για την εξασφάλιση μιας ενιαίας αισθητικής στις διάφορες λειτουργίες του.

Πλεονεκτήματα Bootstrap

- Εύκολο στην εγκατάσταση και στη χρήση: Οποιοσδήποτε με βασικές γνώσεις HTML και CSS μπορεί να το χρησιμοποιήσει.
- Συμβατότητα με όλα τα προγράμματα περιηγητών: Σύστημα εκκίνησης που είναι συμβατό με όλα τα σύγχρονα προγράμματα περιήγησης (Google Chrome, Firefox, Internet Explorer, Safari, και Opera). Μια από τις

προκλήσεις που συναντούνε οι προγραμματιστές είναι ότι μία ιστοσελίδα φαίνεται διαφορετική σε διαφορετικά προγράμματα περιηγητών.

- Συμβατό με κινητές συσκευές: Τις οποίες θέτει σε απόλυτη προτεραιότητα. Η ιστοσελίδα φαίνεται ίδια σε «έξυπνα» τηλέφωνα, υπολογιστές χειρός και επιτραπέζιους υπολογιστές.
- Δημιουργία ευαίσθητων (responsive) ιστοσελίδων: Μπορούμε να κρύψουμε κάποιο περιεχόμενο ανάλογα με το μέγεθος της οθόνης. Για παράδειγμα προσθέτοντας μια κλάση .visible σε ένα στοιχείο, μπορούμε να το κάνουμε ορατό μόνο σε laptop.
- Τεκμηρίωση: Το Bootstrap όχι μόνο προσφέρει μορφοποίηση για σχεδόν κάθε στοιχείο, μια τυπική ιστοσελίδα ή εφαρμογή παρέχει επίσης μια μεγάλη τεκμηρίωση με παραδείγματα που το καθιστούν εύκολο στην εφαρμογή.
- Βασική μορφοποίηση για τα περισσότερα στοιχεία HTML: Μια ιστοσελίδα που έχει πολλά διαφορετικά στοιχεία, όπως επικεφαλίδες, λίστες, πίνακες, κουμπιά, φόρμες, κλπ. Όλα αυτά τα βασικά στοιχεία της HTML έχουν μορφοποιηθεί και ενισχυθεί με επεκτάσιμες κλάσεις.
- Επεκτάσιμη: Διαθέτει JavaScript ενσωματώσεις (plugins).

ΥΠΟΚΕΦΑΛΑΙΟ 3.5 – MySQL/H2

Η **MySQL** είναι ένα ελεύθερο σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων. Τρέχει ως εξυπηρετής (server) και επιτρέπει την πρόσβαση πολλών χρηστών σε πολλές βάσεις δεδομένων. Χρησιμοποιείται σε κάποιες δημοφιλείς υπηρεσίες όπως το Youtube, την Wikipedia, το Google, το Facebook κ.ά. Μια βάση δεδομένων επιτρέπει την αποθήκευση, την αναζήτηση, την ταξινόμηση και την ανάκληση των δεδομένων. Χρησιμοποιεί τις εντολές γνωστές από την SQl απλά σε πιο γραφικό περιβάλλον. Η MySQL έχει πολλά πλεονεκτήματα, όπως χαμηλό κόστος, εύκολη διαμόρφωση και μάθηση και ο κώδικας προέλευσης είναι αποτελεσματικός. Χωρίς την χρήση της MySql ο ιστότοπος είναι ένα υπερκείμενο μεν δηλαδή εκτός από απλό κείμενο θα περιέχει και εικόνες και άλλα πολυμέσα αλλά παρόλα αυτά, οι χρήστες δε θα μπορούν να προσωποποιήσουν την πλοήγηση τους με την δημιουργία και τη επεξεργασία ενός λογαριασμού χρήστη (username – password).

Η **H2** είναι μια βάση δεδομένων ανοικτού κώδικα για Java. Μπορεί να ενσωματωθεί σε εφαρμογές Java ή να εκτελεστεί στη λειτουργία πελάτη-διακομιστή. Η βάση δεδομένων H2 μπορεί να ρυθμιστεί ώστε να λειτουργεί ως βάση δεδομένων στη μνήμη, πράγμα που σημαίνει ότι τα δεδομένα δεν θα παραμείνουν στο δίσκο.

Για λόγους ευκολίας, η H2 χρησιμοποιήθηκε για την ανάπτυξη της πτυχιακής. Η MySQL θα είναι η κύρια τεχνολογία για την διαχείριση βάσης, μονάχα και όταν η διαδικτυακή εφαρμογή στηθεί σε κάποιο server για κανονική χρήση στο ευρύ κοινό.

ΚΕΦΑΛΑΙΟ 4 – ΔΙΑΧΕΙΡΙΣΗ ΣΥΣΤΗΜΑΤΟΣ

ΕΙΣΑΓΩΓΗ

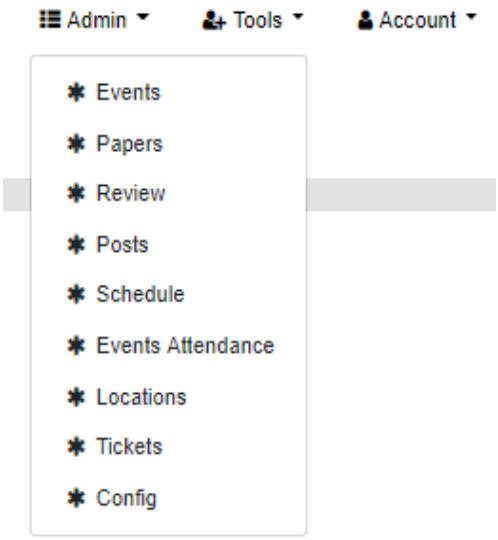
Σε αυτό το κεφάλαιο αναφέρουμε τις βασικές λειτουργίες της εφαρμογής. Παρουσιάζουμε την εφαρμογή με διάφορες εικόνες, παραδείγματα και εκτελέσεις με την ανάλογη επεξήγηση. Στα πρώτα 4 υποκεφάλαια αναλύουμε την ιστοσελίδα από την μεριά του διαχειριστή και πως αυτός μπορεί να δημιουργήσει το αντίστοιχο περιεχόμενο. Στα επόμενα 2 αναλύουμε το πως χρησιμοποιεί την ιστοσελίδα ένας απλός χρήστης και ένας κριτής δημοσιεύσεων.

ΥΠΟΚΕΦΑΛΑΙΟ 4.1 – ΔΙΑΧΕΙΡΙΣΗ ΚΑΙ ΔΗΜΙΟΥΡΓΙΑ ΝΕΟΥ ΣΥΝΕΔΡΙΟΥ

Ο διαχειριστής της διαδικτυακής εφαρμογής είναι αυτός που δημιουργεί την ιστοσελίδα του ηλεκτρονικού συνεδρίου από την αρχή. Με αυτό εννοούμε ότι, όταν ληφθεί η εφαρμογή στα χέρια του διαχειριστή, αυτός θα στήσει, θα εμπλουτίσει και διοργανώσει την εφαρμογή με το περιεχόμενο, ενημερωτικό υλικό, πληροφορίες του συνέδριου κ.α.

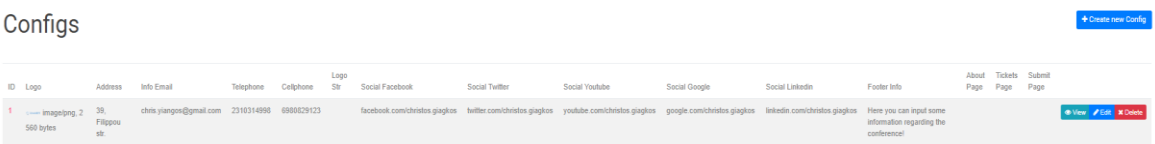
Ως διαχειριστής, κάνοντας login στην εφαρμογή κάτω από το dropdown menu με το όνομα ‘Admin’, θα βρει διάφορα εργαλεία που θα τον βοηθήσουν να στήσει το περιεχόμενο της εφαρμογής αλλά και να εμπλουτίσει το ηλεκτρονικό συνέδριο με υλικό.
Για παράδειγμα, ο διαχειριστής θα πρέπει να δηλώσει, κάτω από την επιλογή ‘Configs’, όλες τις απαραίτητες πληροφορίες του συνεδρίου. Αυτές συμπεριλαμβάνουν τυχόν λογότυπα, διευθύνσεις, στοιχεία επικοινωνίας, διευθύνσεις κλπ.

Παρακάτω φαίνεται το μενού το οποίο βλέπει ο διαχειριστής όταν εισέρχεται στη σελίδα και από το οποίο μπορεί να δημιουργήσει ό,τι επιθυμεί και να κάνει τις απαραίτητες αλλαγές.



Εικόνα 6 - Μενού Διαχείρισης

Όπως ήδη αναφέραμε, το πρώτο βήμα του διαχειριστή είναι να δώσει πληροφορίες για αυτό το συνέδριο κάτω από την επιλογή 'Config'. Πατώντας το κουμπί με την ένδειξη 'Create New Config' ανοίγει ένα pop up με τα κατάλληλα πεδία προς συμπλήρωση. Συμπληρώνοντας αυτά τα πεδία και ύστερα αποθηκεύοντας τα, θα δούμε ότι το υλικό θα συμπληρωθεί στη σελίδα στα κατάλληλα σημεία όπως για παράδειγμα, τα στοιχεία επικοινωνίας που βρίσκονται στο footer της ιστοσελίδας μας.



Εικόνα 7 - Πίνακας διαχείρισης ιστοσελίδας

Υπάρχει πάντα η δυνατότητα της ενημέρωσης αυτών, πατώντας στο κουμπί με την ένδειξη ‘Edit’.

Εικόνα 8 - Footer

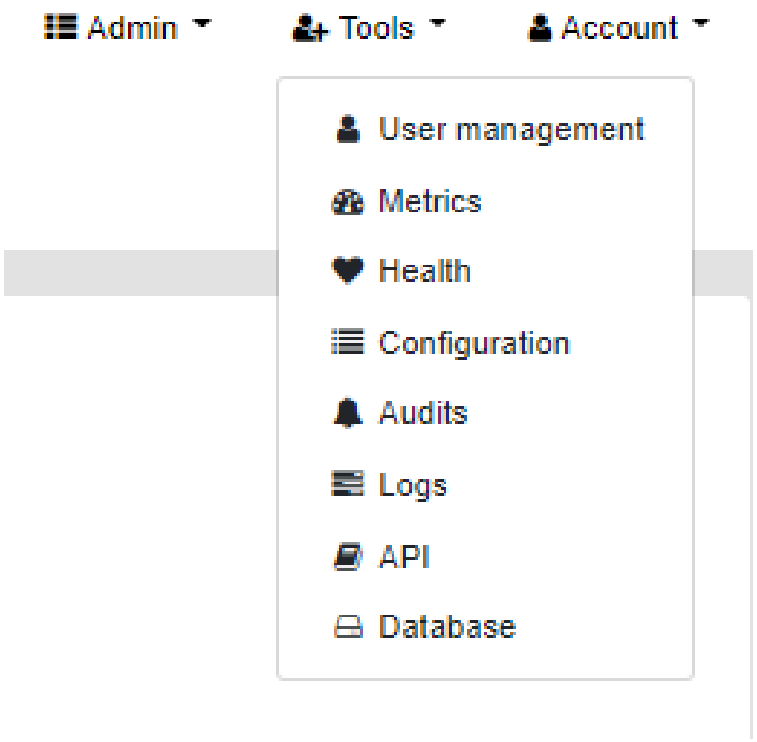
Ο διαχειριστής, πέρα από αρχική περισυλλογή και δημιουργία των γενικών πληροφοριών της ιστοσελίδας, έχει πρόσβαση σε όλα τα δεδομένα αυτού, όπως είναι οι δημοσιεύσεις, οι εγγεγραμμένοι χρήστες, οι αναθέσεις δημοσιεύσεων σε κριτές, τα νέα κλπ.

Όλα τα προαναφερθέντα βρίσκονται και αυτά κάτω από την επιλογή ‘Admin’, όπου βρίσκεται η επιλογή ‘Config’. Για αυτά θα κάνουμε λόγο στα επόμενα υποκεφάλαια.

Επιπλέον, δίνεται στο διαχειριστή η επιλογή να παρακολουθεί, μέσω διαφόρων εργαλείων, την κατάσταση της ιστοσελίδας. Αυτά είναι:

- User Management: Μπορεί να παρακολουθήσει τα στοιχεία των εγγεγραμμένων χρηστών και την κατάσταση του λογαριασμού τους (active, inactive).
- Metrics: Μπορεί να δει την κατάσταση του δίσκου, της μνήμης, της βάσης κλπ όπου η ιστοσελίδα φιλοξενείται.
- Configuration
- Audits: Μπορεί να ελέγξει ποιοι χρήστες αλληλοεπίδρασαν με τον ιστοτόπο, όπως είναι οι εγγραφές, οι είσοδοι, κλπ.
- Logs: Μπορεί να ενεργοποιήσει κάποιες λειτουργίες ώστε να παράγουν επιπλέον δεδομένα στα logs του συστήματος για να μπορέσει, πιθανόν, να κάνει κάποιο debugging.
- API: Μπορεί να δει τα διαθέσιμα endpoints που παρέχονται στην εφαρμογή.

- Database: Μπορεί να δει τα δεδομένα μέσα στην βάση του συστήματος με τη δυνατότητα εκτέλεσης διαφόρων SQL Queries.



Εικόνα 9 - Εργαλεία κατάστασης ιστοσελίδας

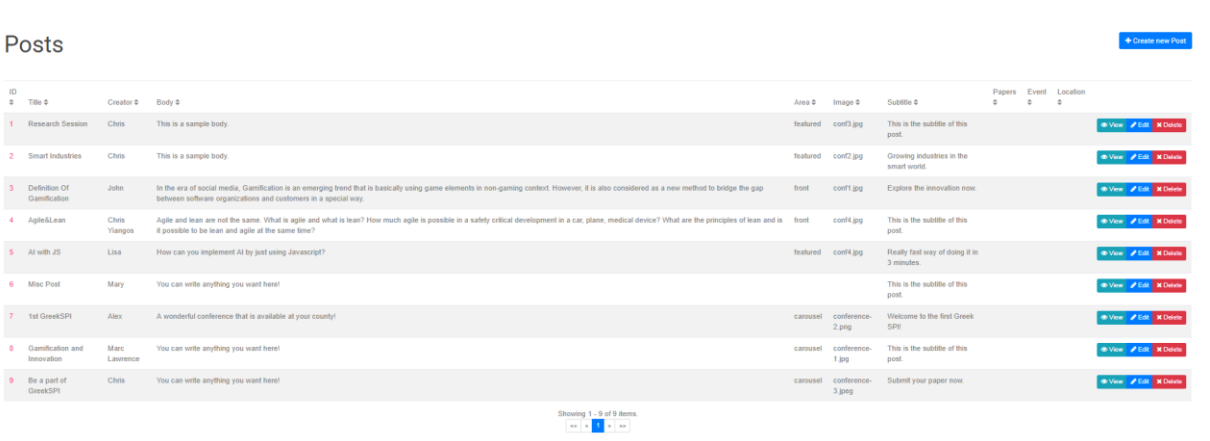
ΥΠΟΚΕΦΑΛΑΙΟ 4.2 – ΔΙΑΧΕΙΡΙΣΗ ΚΑΙ ΔΗΜΙΟΥΡΓΙΑ ΝΕΩΝ ΑΡΘΡΩΝ

Η διαχείριση και δημιουργία άρθρων αποτελεί την περιοχή μέσα στην οποία μπορεί να γίνει η διαχείριση όλων των άρθρων που πρόκειται να χρησιμοποιηθούν και δημοσιευθούν στην ιστοσελίδα.

Όπως ήδη αναφέραμε, ο διαχειριστής μπορεί, μέσω των εργαλείων που του παρουσιάζονται, να δημιουργήσει άρθρα στην ιστοσελίδα. Για να ξεκινήσει την διαδικασία δημιουργίας απλά επιλέγει ‘Posts’ από τα διαθέσιμα εργαλεία που του δίνονται κάτω από το dropdown ‘Admin’.

Έτσι, όταν η σελίδα φορτώσει θα βλέπει μια λίστα με τα ήδη δημιουργημένα άρθρα και θα του δίνεται η δυνατότητα να επεξεργαστεί τα υπάρχοντα, ή να δημιουργήσει καινούρια, πατώντας το κουμπί με την ένδειξη ‘Create New’.

Στην παρακάτω εικόνα βλέπουμε μια λίστα από ήδη υπάρχουσες δημοσιεύσεις.



Εικόνα 10 - Πίνακας άρθρων

Οι επιλογές που έχει στην διάθεση του οπ διαχειριστής για την δημιουργία ενός άρθρου είναι οι εξής:

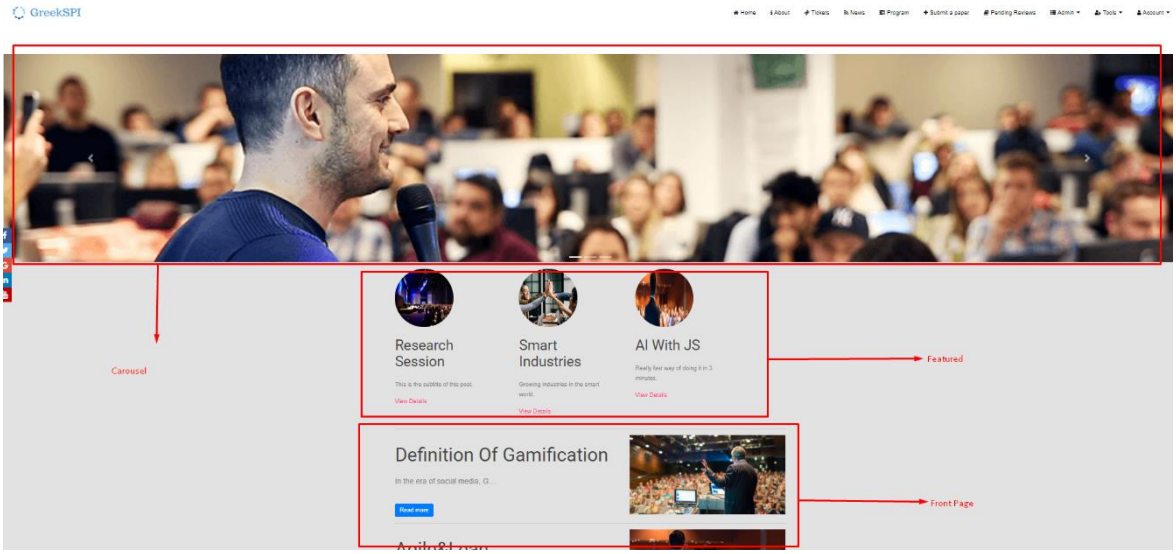
- Title: Όπου προσδιορίζει τον κεντρικό τίτλο του άρθρου (Απαραίτητο)
- Creator: Ο δημιουργός/συντάκτης του άρθρου.
- Body: Το περιεχόμενο του άρθρου.

- Area: Σε αυτό το πεδίο ορίζουμε σε ποιο σημείο θα θέλουμε να προβάλλεται το άρθρο μας στην ιστοσελίδα. Οι επιλογές που έχουμε είναι τρεις: α) Featured, β) Front Page, γ) Carousel. Κάθε μια από τις επιλογές καθορίζει ουσιαστικά σε ποια σελίδα και σε ποιο σημείο της σελίδας θα προβληθεί το άρθρο.

  Για παράδειγμα, το Area : Carousel θα προβάλει το άρθρο στην αρχική σελίδα στην περιοχή των κυλιόμενων εικόνων.

  Το Area: Front Page θα προβάλει το άρθρο στη λίστα των άρθρων στην αρχική σελίδα.

  Το Area: Featured θα προβάλει το άρθρο στην αρχική σελίδα κάτω από το carousel με διαφορετικό θέμα από τα υπόλοιπα άρθρα τα οποία τα κάνει να ξεχωρίζουν με την ιδιαιτερότητα που έχουν στην εμφάνιση των εικόνων τους.



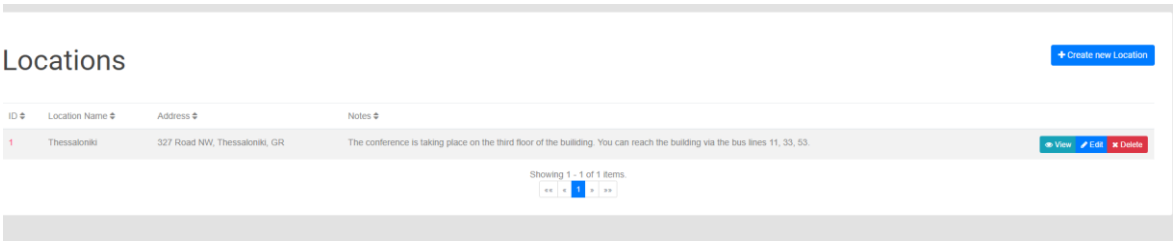
Εικόνα 11 - Θέσεις προβολής άρθρων

- Image: Η εικόνα που θα χρησιμοποιηθεί για το άρθρο
- Subtitle: Ο υπότιτλος που θα εμφανίζεται στην προεπισκόπηση του άρθρου
- Paper: Προαιρετικό. Αν κάποιο άρθρο συσχετίζεται με δημοσίευση κάποιου χρήστη, μας δίνεται η επιλογή να μπορέσουμε να συνδέσουμε άμεσα τη δημοσίευση με το άρθρο ώστε να μπορέσουν οι αναγνώστες να διαβάσουν τη δημοσίευση.

ΥΠΟΚΕΦΑΛΑΙΟ 4.3 – ΔΙΑΧΕΙΡΙΣΗ ΚΑΙ ΔΗΜΙΟΥΡΓΙΑ ΠΡΟΓΡΑΜΜΑΤΟΣ ΣΥΝΕΔΡΙΟΥ

Ο διαχειριστής είναι αυτός που θα δημιουργήσει το τρέχον συνέδριο, θέτοντας το όνομα, την ημερομηνία αλλά και την τοποθεσία του συνεδρίου.

Πρώτο του βήμα στη δημιουργία του συνεδρίου είναι να ορίσει την τοποθεσία οπού θα διεξαχθεί το συνέδριο. Αυτό γίνεται επιλέγοντας να δημιουργήσει μια νέα τοποθεσία κάτω από την επιλογή ‘Locations’. Εκεί, στα αντίστοιχα πεδία, θέτει τις απαραίτητες πληροφορίες όπως την πόλη, την περιοχή, το όνομα του κτιρίου, τη διεύθυνση αλλά και διάφορες σημειώσεις όπως την ακριβή αίθουσα και τον όροφο όπου θα λάβει χώρα το συνέδριο.

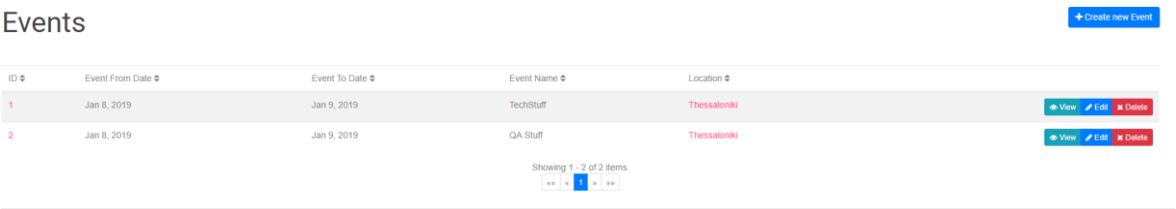


Εικόνα 12 - Πίνακας τοποθεσιών

Το επόμενο βήμα είναι η δημιουργία του ίδιου του συνεδρίου. Για να επιτευχθεί αυτό ο διαχειριστής θα πρέπει να δημιουργήσει, κάτω από την επιλογή ‘Events’, μια νέα εγγραφή με τα απαραίτητα στοιχεία που χρειάζονται. Αυτά είναι τα εξής:

- Event From Date: Η ημερομηνία έναρξης του συνεδρίου
- Event To Date: Η ημερομηνία λήξης του συνεδρίου. Αν το συνέδριο διαρκεί μία ημέρα τότε αυτό το πεδίο μπορεί να παραμείνει κενό.
- Event Name: Το όνομα του συνέδριου. Εδώ, να σημειωθεί, ότι αν το συνέδριο μας ακολουθεί Tracks μπορούμε να δημιουργήσουμε πολλαπλές εγγραφές και να θέσουμε στην επιλογή αυτή το όνομα του track. Ειδάλλως, δημιουργούμε μονάχα μια εγγραφή και θέτουμε απλά το όνομα του συνεδρίου.

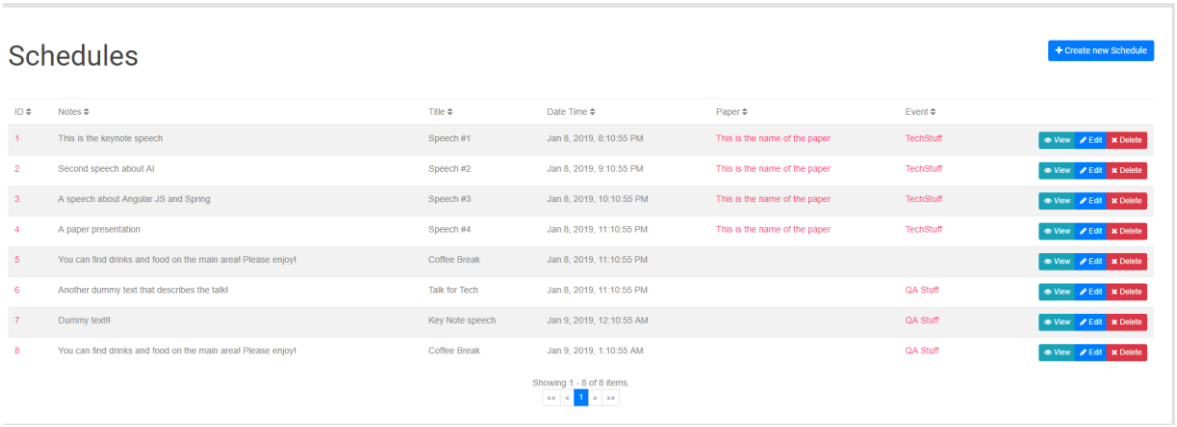
- Location: Σε αυτό το πεδίο εμφανίζονται οι διαθέσιμες πόλεις/περιοχές που θέσαμε στο προηγούμενο βήμα, σε μορφή dropdown.



Εικόνα 13 - Πίνακας ονομάτων/θεμάτων συνεδρίου

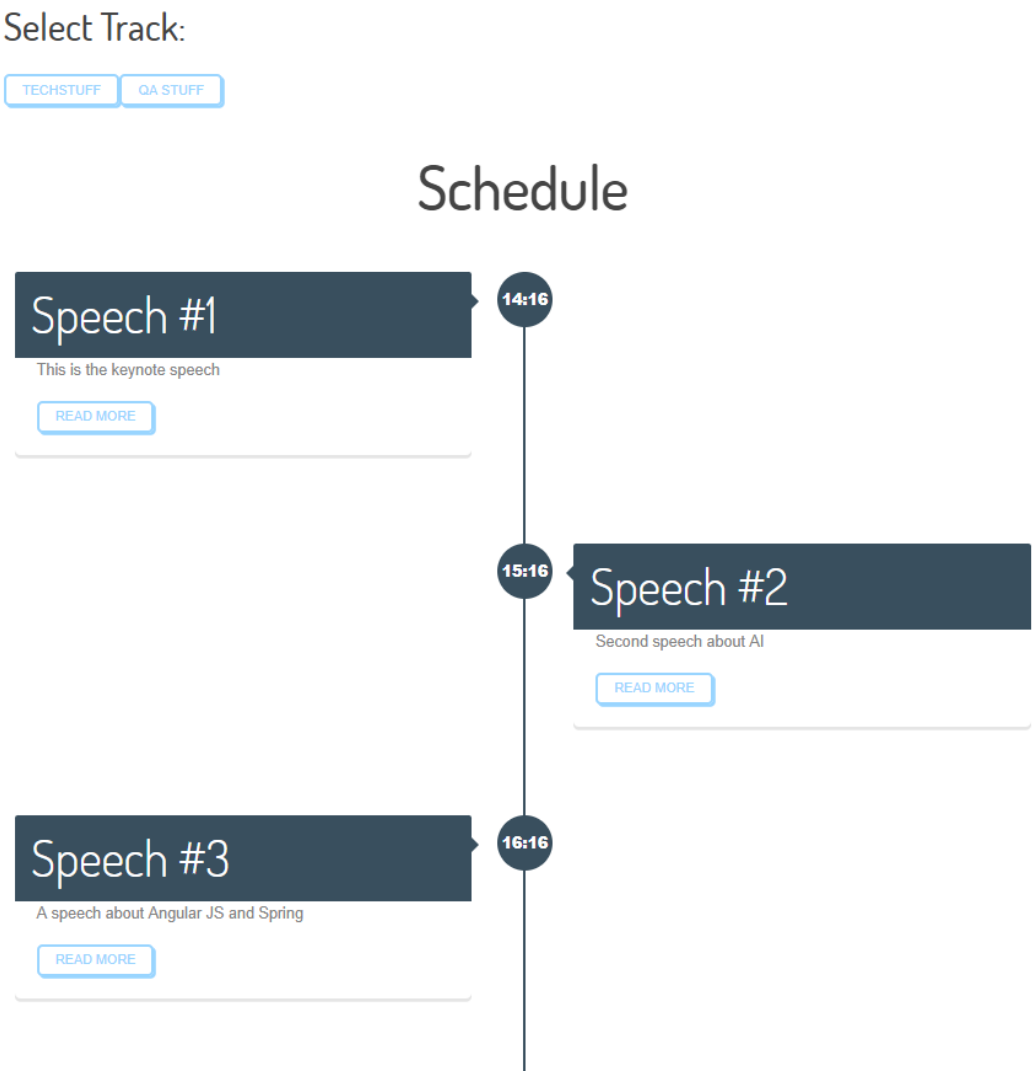
Μια επιπλέον ευθύνη του διαχειριστή του συστήματος είναι η δημιουργία και συντήρηση του ορθού χρονολογικού προγράμματος του συνεδρίου. Κάτω από την επιλογή ‘Schedule’ μπορεί να δημιουργήσει και να επεξεργαστεί το πρόγραμμα του συνεδρίου.
Τα διαθέσιμα πεδία προς συμπλήρωση για τη δημιουργία ενός ορθού συνεδρίου είναι τα εξής:

- Title: Ο τίτλος της ομιλίας/παρουσίασης της δημοσίευσης.
- Date/Time: Η ημέρα και η ώρα της ομιλίας/παρουσίασης.
- Paper: Η συσχετιζόμενη δημοσίευση.
- Event: Το συνέδριο ή το track στο οποίο παρουσιάζεται η ομιλία.
- Notes: Λοιπές σημειώσεις που αφορούν την ομιλία, όπως μια εισαγωγή στο θέμα της παρουσίασης.



Εικόνα 14 - Πίνακας προγράμματος

Το τελικό αποτέλεσμα που θα βλέπει ένας οποιοσδήποτε χρήστης που θα επισκεφθεί την σελίδα, είναι ένα « χρονολόγιο » όπου θα παρουσιάζονται οι τίτλοι των ομιλιών και οι σημειώσεις πάνω σε αυτές.
Στην περίπτωση που το συνέδριο ακολουθεί ‘tracks’ θα δίνεται η δυνατότητα στο χρήστη να επιλέξει το αντίστοιχο track ώστε να δει τις αντίστοιχες ομιλίες.



Εικόνα 15 - Παράσταση προγράμματος συνεδρίου στην μορφή timeline

Όπως βλέπουμε στην παραπάνω εικόνα, στο επάνω αριστερά μέρος δίνεται η δυνατότητα επιλογής track και από κάτω εμφανίζεται με λεπτομέρειες το πρόγραμμα του συνεδρίου για το επιλεγμένο track.

ΥΠΟΚΕΦΑΛΑΙΟ 4.4 – ΔΙΑΧΕΙΡΙΣΗ ΚΑΙ ΔΗΜΙΟΥΡΓΙΑ REVIEWS (ΔΙΑΧΕΙΡΙΣΤΗΣ)

Το βασικό και καινοτόμο χαρακτηριστικό της ιστοσελίδας είναι τα ‘Reviews’. Τα reviews στην προκειμένη είναι οι αναθέσεις ορισμένων δημοσιεύσεων από απλούς χρήστες, σε χρήστες όπου έχουν τον ρόλο του κριτή.

Ένας χρήστης ο οποίος έχει τον ρόλο του κριτή, βασικό του καθήκον είναι να εξετάσει και να αξιολογήσει τη δημοσίευση ενός απλού χρήστη και να την εγκρίνει ή να την απορρίψει, αλλά ακόμα και να την απορρίψει με προτεινόμενες διορθώσεις ώστε ο χρήστης να τη διορθώσει και να την καταθέσει εκ νέου. Ο ρόλος του κριτή και οι διαδικασίες που ακολουθεί αναλύονται στα παρακάτω υποκεφάλαια.

Ας δούμε τώρα πως διαχειριζόμαστε τις δημοσιεύσεις των χρηστών, πως αναθέτουμε τις δημοσιεύσεις στους κριτές και τις διάφορες λειτουργίες αυτού του χαρακτηριστικού της ιστοσελίδας.

Όταν ο διαχειριστής εισέρχεται στην ιστοσελίδα εμφανίζεται το κουμπί με την ένδειξη ‘Pending Reviews’. Αφότου επισκεφτούμε αυτήν τη σελίδα, βλέπουμε τις εξής τέσσερις ενότητες:

- Pending Reviews
- All Reviews
- All Papers
- Unassigned Papers

Κάθε μία από αυτές τις ενότητες έχει και τις αντίστοιχες εγγραφές, σε μορφή πίνακα με τα αντίστοιχα κελιά να περιέχουν τις απαραίτητες πληροφορίες.

Ας πάμε τώρα να αναλύσουμε τις ενότητες αυτές και τις λειτουργίες τους.

1. Pending Reviews

Σε αυτήν την ενότητα ο διαχειριστής βλέπει μια εγγραφή με τη δημοσίευση που του έχει ανατεθεί να κρίνει. Συγκεκριμένα, η εγγραφή έχει τις εξής πληροφορίες:

- Paper Title: Τον τίτλος της δημοσίευσης.
- Authors: Τα ονόματα των δημιουργών της δημοσίευσης.
- Paper: Η δημοσίευση σε μορφή μεταφορτώσιμου αρχείου.
- Username: Ο χρήστης που μεταφόρτωσε τη δημοσίευση.
- Notes: Σε αυτό το πεδίο δίνεται η επιλογή στο διαχειριστή να μεταφορτώσει ένα αρχείο το οποίο μπορεί να περιέχει σημειώσεις που αφορούν τη δημοσίευση, όπως είναι τυχόν διορθώσεις, κ.α.
- Status: Σε αυτό το πεδίο εμφανίζεται η κατάσταση στην οποία βρίσκεται η δημοσίευση τη συγκεκριμένη χρονική στιγμή. Επίσης υπάρχουν και τα εξής κουμπιά με τις ενδείξεις “Approve” και “Review Failed”. Πατώντας αυτά τα κουμπιά αλλάζει η κατάσταση της δημοσίευσης/κριτικής και ο χρήστης ενημερώνεται για αυτό.

Θα μιλήσουμε αναλυτικότερα παρακάτω για την διαδικασία με την οποία γίνεται η κριτική μιας δημοσίευσης, από την πλευρά του κριτή.

2. All Reviews

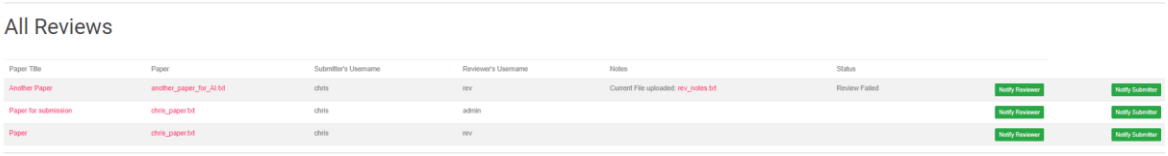
Στην ενότητα αυτή περιλαμβάνονται όλες οι κριτικές που έχουν ανατεθεί σε όλους του κριτές της ιστοσελίδας, εμφανίζοντας επίσης και την κατάσταση στην οποία βρίσκονται τα τρέχον μεταφορτωμένα αρχεία, κ.λπ.

Τα πεδία τα οποία βρίσκονται κάτω από αυτήν την ενότητα είναι τα εξής:

- Paper Title: Ο τίτλος της δημοσίευσης.
- Paper: Η δημοσίευση σε μορφή μεταφορτώσιμου αρχείου.
- Submitter’s Username: Το username του χρήστη που μεταφόρτωσε τη δημοσίευση.
- Reviewer’s Username: Το username του χρήστη, όπου έχει το ρόλο του κριτή, στον οποίο έχει ανατεθεί η δημοσίευση.
- Notes: Το μεταφορτωμένο αρχείο το οποίο έχει ανεβάσει ο κριτής
- Status: Η τρέχουσα κατάσταση της κριτικής.
- Δύο κουμπιά με τις ενδείξεις “Notify Reviewer” και “Notify Submitter”.

Ο διαχειριστής, με τη βοήθεια αυτών των πεδίων κάτω από αυτήν την ενότητα μπορεί εύκολα και γρήγορα να δει οποιαδήποτε στιγμή την κατάσταση των ανατεθειμένων δημοσιεύσεων. Μπορεί επίσης να διαβάσει τις σημειώσεις του κριτή αλλά και την ίδια τη δημοσίευση.

Επίσης, του δίνεται η δυνατότητα να ειδοποιήσει είτε τον κριτή, είτε τον δημιουργό για την κατάσταση στην οποία βρίσκεται η κριτική, ώστε να προχωρήσει η διαδικασία. Αυτό γίνεται εύκολα απλά πατώντας το αντίστοιχο κουμπί. Πατώντας το κουμπί, ο ανάλογος χρήστης, θα λάβει στο προσωπικό του λογαριασμό ηλεκτρονικού ταχυδρομείου ένα e-mail, όπου θα αναφέρεται η κατάσταση στην οποία βρίσκεται η κριτική.



Εικόνα 16 - Πίνακας διαθέσιμων κριτικών

3. All papers

Στην ενότητα αυτή οι εγγραφές συμπεριλαμβάνουν τα εξής πεδία:

- Paper Name: Το όνομα της δημοσίευσης.
- Paper: Η δημοσίευση σε μορφή μεταφορτώσιμου αρχείου.
- Authors: Τα ονόματα των συγγραφέων της δημοσίευσης.
- Username: Ο χρήστης που μεταφόρτωσε την δημοσίευση.

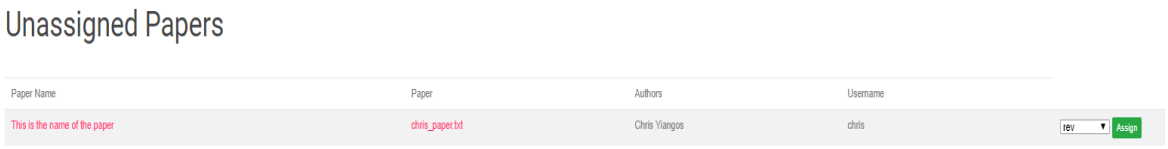
Οι εγγραφές αυτές δείχνουν όλες τις δημοσιεύσεις οι οποίες έχουν ανέβει στην ιστοσελίδα, με σκοπό να περάσουν από κριτική και να παρουσιαστούν στο συνέδριο.

4. Unassigned Papers

Κάτω από την ενότητα Unassigned Papers δίνεται η επιλογή σε ένα διαχειριστή να δει όλες τις δημοσιεύσεις οι οποίες δεν έχουν ανατεθεί σε κάποιον κριτή. Τα πεδία που περιλαμβάνονται είναι τα εξής:

- Paper Name: Το όνομα της δημοσίευσης.
- Paper: Η δημοσίευσης σε μορφή μεταφορτώσιμου αρχείου.
- Authors: Τα ονόματα των συγγραφέων της δημοσίευσης.
- Username: Ο χρήστης που μεταφόρτωσε την δημοσίευση.
- Ένα dropdown με τη λίστα των διαθέσιμων κριτών του συστήματος και ένα κουμπί με την ένδειξη ‘Assign’.

Βλέποντας αυτές τις εγγραφές, ένας διαχειριστής μπορεί να αποφασίσει, εύκολα και γρήγορα, σε ποιον κριτή θα αναθέσει μια δημοσίευση για να την αξιολογήσει. Το μόνο που έχει να κάνει είναι να επιλέξει τον ανάλογο κριτή, από το dropdown που βρίσκεται στην ίδια σειρά με τη δημοσίευση, και να πατήσει το κουμπί με την ένδειξη ‘Assign’.
Έτσι, αυτόματα, η δημοσίευση θα έχει ανατεθεί σε κάποιο χρήστη με τον ρόλο του κριτή. Αμέσως ο κριτής θα ενημερωθεί με ένα μήνυμα ηλεκτρονικού ταχυδρομείου και θα αποφασίσει αν θα προχωρήσει με την κριτική της δημοσίευσης. Περισσότερα για τα βήματα που ακολουθεί ένας κριτής, αναλύονται παρακάτω.



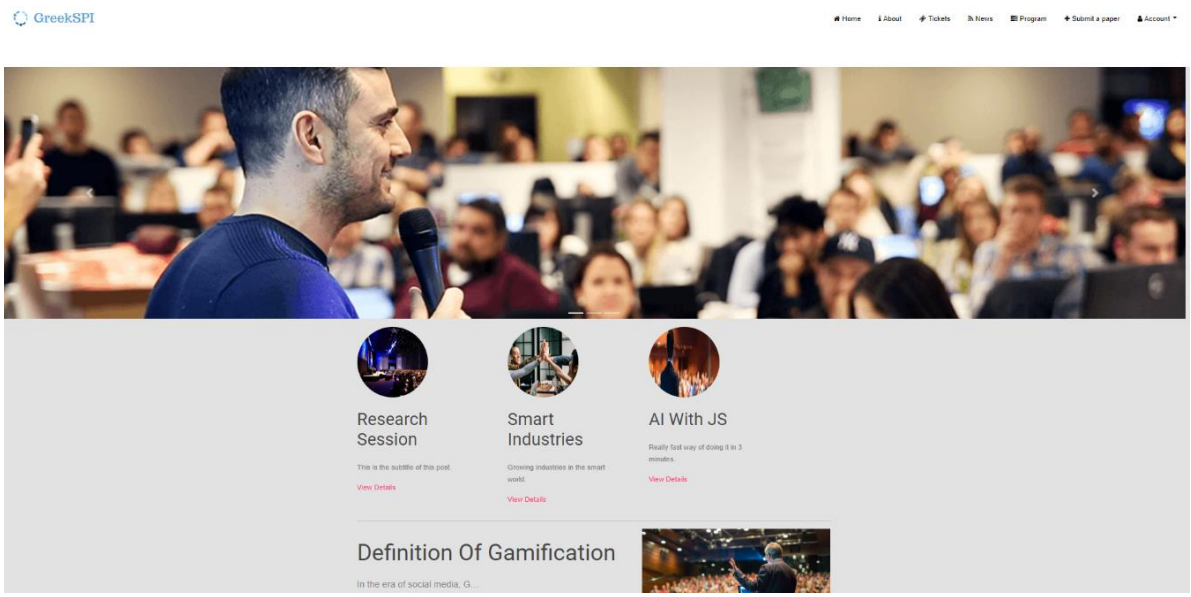
Εικόνα 17 - Πίνακας μη-ορισμένων κριτικών

ΥΠΟΚΕΦΑΛΑΙΟ 4.5 – ΔΗΜΙΟΥΡΓΙΑ ΔΗΜΟΣΙΕΥΣΗΣ (ΧΡΗΣΤΗΣ)

Η βασική πηγή για τον εμπλουτισμό και την ορθή χρησιμότητα της διαδικτυακής αυτής εφαρμογής, είναι ο απλός εγγεγραμμένος χρήστης που επρόκειτο να καταθέσει την δημοσίευση του, με κύριο σκοπό τη συμμετοχή του στο συνέδριο.

Οι χρήστες του συστήματος, πέρα από την ενημέρωση που έχουν στα χέρια του, έχουν την δυνατότητα, όπως ήδη προαναφέραμε, να καταθέσουν τη δική τους δημοσίευση στην ιστοσελίδα, ώστε να περάσει τη διαδικασία της κριτικής, με αποτέλεσμα και απώτερο σκοπό την παρουσίαση/δημοσίευση αυτής στο συνέδριο.

Ας δούμε όμως, πως ένα επισκέπτης κάνει εγγραφή στην ιστοσελίδα και πως καταθέτει τη δική του δημοσίευση με σκοπό τη συμμετοχή του στο συνέδριο.



Εικόνα 18 - Αρχική σελίδα

Επισκέπτοντας για πρώτη φορά την ιστοσελίδα, ο χρήστης παρατηρεί στην αρχική σελίδα διάφορα άρθρα που σχετίζονται με το συνέδριο τους φορείς, τα θέματα που θα αναλυθούν κ.α. Στο header της ιστοσελίδας έχει όλους τους διαθέσιμους υπερσυνδέσμους που βοηθούν στην περιήγηση αυτής.

Αναλύονται οι υπερσύνδεσμοι, το περιεχόμενό τους και οι λειτουργίες τους:

- Home

- About
- Tickets
- News
- Program
- Submit a paper
- Account (Login/Register)

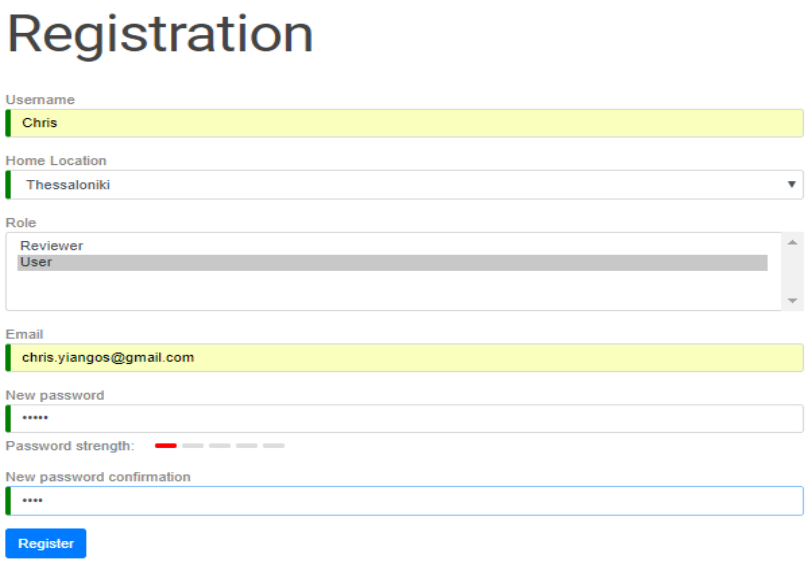
Οι παραπάνω σελίδες αναλύονται στο υποκεφάλαιο 5.7

Ένας εγγεγραμμένος χρήστης βλέπει, στο header της ιστοσελίδας, βλέπει πέρα από τα παραπάνω και ακόμη μια επιλογή με το όνομα ‘My Papers’. Κάτω από αυτήν την επιλογή ο χρήστης έχει την δυνατότητα να καταθέσει την έρευνα του ώστε να συμμετάσχει στο επερχόμενο συνέδριο.

Ας δούμε όμως αναλυτικότερα πως ένας επισκέπτης εγγράφεται στην ιστοσελίδα και καταθέτει τη δημοσίευση του σε αυτήν.

1. Εγγραφή (Register)

   Ο χρήστης, πατώντας πάνω στην ένδειξη ‘Account’, έχει δύο επιλογές, είσοδος ή εγγραφή (Login/Register).
   Πατώντας στην ένδειξη ‘Register’, εμφανίζεται η φόρμα εγγραφής όπου και πρέπει να συμπληρωθεί ορθά για την επίτευξη της εγγραφής



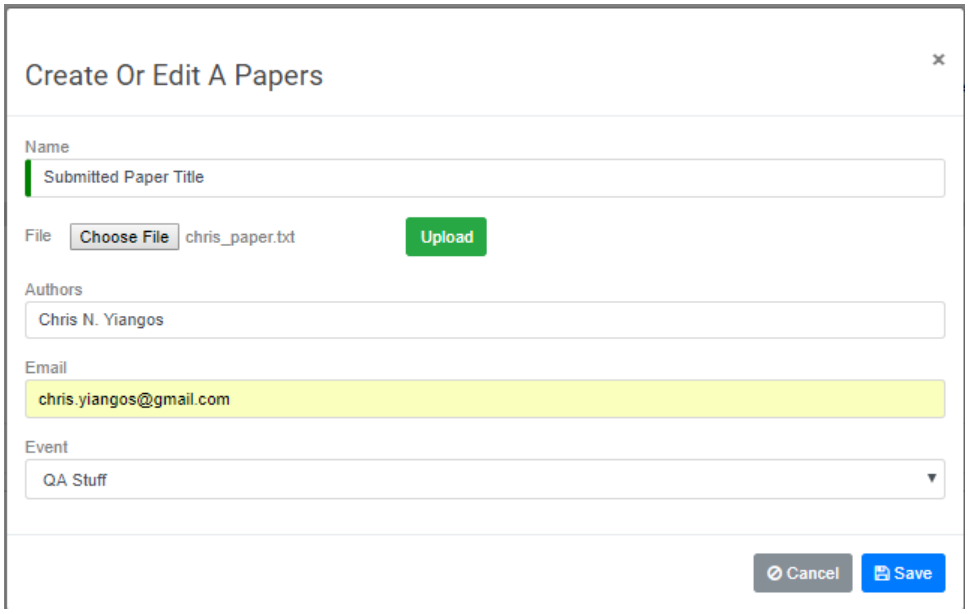
Εικόνα 19 - Φόρμα εγγραφής

Στη φόρμα αυτή δίνεται η επιλογή στο χρήστη να επιλέξει το ρόλο του ανάμεσα σε αυτόν του κριτή και του απλού χρήστη. Στη συγκεκριμένη περίπτωση μιλάμε για ένα χρήστη ο οποίος θα καταθέσει μια δημοσίευση, όποτε η δική μας επιλογή είναι αυτή του απλού χρήστη (επιλογή User).
Εφόσον η εγγραφή μας είναι επιτυχημένη ο χρήστης θα λάβει ένα email που θα περιέχει έναν υπερσύνδεσμο για επιβεβαίωση στοιχείων. Ακολουθώντας αυτόν τον υπερσύνδεσμο, ο χρήστης θα μεταφερθεί στην ιστοσελίδα και θα ενημερωθεί για την ορθή δημιουργία του λογαριασμού του.

2. Κατάθεση Δημοσίευσης (Submit Paper)

Αφότου ο χρήστης έχει ολοκληρώσει την εγγραφή του και είναι συνδεδεμένος με τον προσωπικό του λογαριασμό, μπορεί πλέον να καταθέσει την ερευνά του, ώστε να προχωρήσει στη διαδικασία της κριτικής. Ένας συνδεδεμένος χρήστης, στο header της ιστοσελίδας βλέπει την ένδειξη ‘My Papers’. Επισκέπτοντας τη σελίδα αυτή, ο χρήστης βλέπει τις συνολικές έρευνες που έχει καταθέσει, τις έρευνες οι οποίες έχουν προχωρήσει σε κριτική ή βρίσκονται σε κάποιο ενδιάμεσο στάδιο κριτικής.

Ο χρήστης για να προχωρήσει σε κατάθεση, θα πρέπει να πατήσει στο κουμπί με την ένδειξη ‘Submit your paper’. Πατώντας το κουμπί αυτό, θα του παρουσιαστεί μια φόρμα την οποία και θα πρέπει να συμπληρώσει και να καταθέσει το αρχείο του το οποίο θα περιέχει και την έρευνα του. Σε αυτήν τη φόρμα μπορεί επίσης να επιλέξει σε ποιο συνέδριο ή track θέλει να ενταχθεί.



Εικόνα 20 - Φόρμα κατάθεσης έρευνας

Αποθηκεύοντας την καταχώρησή του, αμέσως ο χρήστης μπορεί να δει την εγγραφή στον πίνακα ‘My Submitted Papers’.

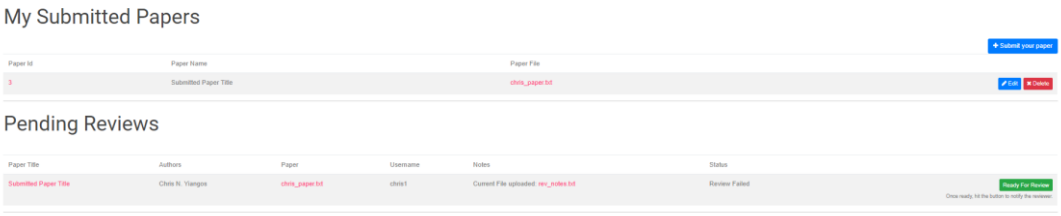
3. Κριτική Έρευνας

Στο ενδιάμεσο, ένας διαχειριστής της ιστοσελίδας αναθέτει την έρευνα του χρήστη σε κάποιον κριτή. Πλέον ο χρήστης βλέπει άλλη μια εγγραφή στον πίνακα ‘Pending Reviews’. Η διαδικασία κριτικής αναλύεται στο επόμενο υποκεφάλαιο.

Αφότου η έρευνα έχει ανατεθεί σε έναν κριτή, υποθέτουμε ότι ο κριτής έχει υποδείξει κάποια σημεία της έρευνας προς διόρθωση. Αυτό σημαίνει ότι η εγγραφή αυτή είναι πλέον σε ‘Review Failed’ κατάσταση. Ο χρήστης έχει

ενημερωθεί με αντίστοιχο email για αυτήν.

Είναι πλέον σειρά του χρήστη να συνδεθεί στην ιστοσελίδα και να διαβάσει το αρχείο που έχει ανεβάσει ο κριτής, κάτω από τον πίνακα ‘Pending Reviews’.



Εικόνα 21 - Πίνακες διαθέσιμη προς τον χρήστη (Κατατεθειμένες έρευνες / εκκρεμής κριτικές)

Ο χρήστης διαβάζει τις σημειώσεις του κριτή και κάνει τις ανάλογες διορθώσεις.

Επόμενο βήμα του χρήστη είναι η κατάθεση της διορθωμένης έρευνας. Πατώντας την ένδειξη ‘Edit’ στην ανάλογη εγγραφή, ο χρήστης μεταφορτώνει τη διορθωμένη έκδοση της έρευνας και πατάει το κουμπί με την ένδειξη ‘Ready For Review’. Πατώντας το κουμπί αυτό, θα ενημερωθεί ο κριτής στο προσωπικό ηλεκτρονικό λογαριασμό ταχυδρομείου και θα ακολουθήσει την διαδικασία από την αρχή.

Ο κριτής αποδέχεται την έρευνα και ο χρήστης ενημερώνεται με αντίστοιχο email για αυτό. Ο χρήστης μπορεί να δει την κατάσταση της εγγραφής του κάτω από τον πίνακα ‘Pending Reviews’, να έχει τεθεί ως ‘Approved’.

Η έρευνα του χρήστη είναι πλέον αποδεκτή από το συνέδριο και έτοιμη προς παρουσίαση σε αυτό.

ΥΠΟΚΕΦΑΛΑΙΟ 4.6 – ΕΚΠΟΝΗΣΗ ΚΡΙΤΙΚΗΣ (ΚΡΙΤΗΣ)

Αφού αναλύσαμε στο προηγούμενο υποκεφάλαιο πως ένας χρήστης καταθέτει την έρευνα του στο συνέδριο, ας δούμε τι γίνεται στο ενδιάμεσο στάδιο από την κατάθεση μέχρι την αποδοχή αυτής, από την μεριά ενός κριτή.

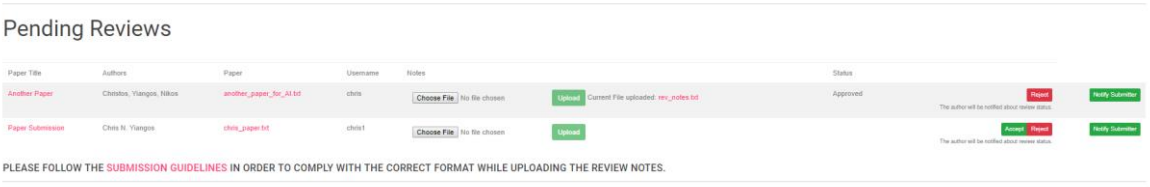
Έχοντας ήδη αναλύσει τη διαδικασία εγγραφής και κατάθεσης έρευνας στην ιστοσελίδα ως χρήστης, τώρα θα αναλύσουμε τις ενέργειες που ακολουθεί ένας χρήστης με τον ρόλο του κριτή για την εκπόνηση της κριτικής.

1. Είσοδος (Login)

   Ο κριτής, έχοντας ήδη δημιουργήσει έναν λογαριασμό στην ιστοσελίδα, ενημερώνεται μέσω email όταν του ανατεθεί μια έρευνα, για να κάνει την κριτική του πάνω σε αυτή.

   Μόλις λάβει το email, συνδέεται με τα στοιχεία του στην ιστοσελίδα και πατάει στο κουμπί με την ένδειξη ‘Pending Review’, στο header της ιστοσελίδας. Σε αυτήν την σελίδα, ο κριτής, βλέπει όλες τις κριτικές που του έχουν ανατεθεί.

   Η τελευταία έρευνα που του έχει ανατεθεί είναι αυτή που δεν θα έχει τεθεί η κατάσταση της και θα δίνεται η δυνατότητα στον κριτή να επιλέξει αν θα τη δεχτεί πατώντας τα κουμπιά ‘Accept’/’Reject’ ανάλογα με την δική του κρίση.



Εικόνα 22 - Εκκρεμής κριτικές - 1

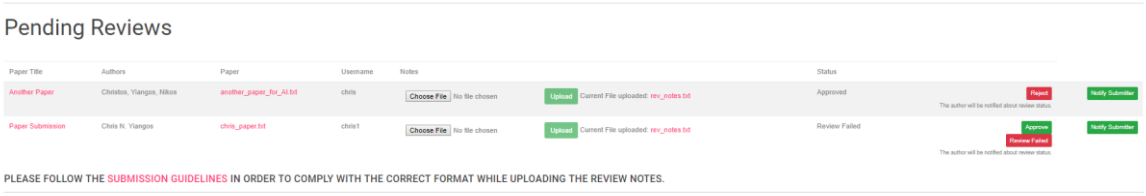
2. Διαδικασία Κριτικής

Αφότου ο κριτής δεχτεί την κριτική πατώντας το κουμπί ‘Accept’, θα πρέπει να διαβάσει την έρευνα και να κρίνει αν είναι δυνατή η παρουσίαση αυτής στο συνέδριο με βάση τα κριτήρια του συνεδρίου.

Αν τη δεχτεί, πατάει στο κουμπί ‘Approve’. Ο χρήστης ενημερώνεται και η έρευνα είναι έτοιμη για παρουσίαση στο συνέδριο.
Αν την απορρίψει λόγω ελλείψεων ή και λαθών σε αυτή, έχει δύο επιλογές:

- Απόρριψη: Ο κριτής απορρίπτει τελείως την έρευνα και δε συμμετέχει στο συνέδριο. Ο κριτής πατάει το κουμπί με την ένδειξη ‘Reject’ και η κατάσταση της κριτικής θέτεται σε ‘Rejected’.

- Απόρριψη με διορθώσεις: Ο κριτής απορρίπτει την έρευνα και ανεβάζει σημειώσεις στο πεδίο ‘Notes’, ώστε ο χρήστης να διορθώσει την έρευνα του και να την ξανάκαταθέσει ώστε να ακολουθηθεί η διαδικασία από την αρχή. Ύστερα, πατώντας το κουμπί με την ένδειξη ‘Review Failed’, ο χρήστης ενημερώνεται και η κατάσταση της κριτικής θέτεται σε ‘Review Failed’.



Εικόνα 23 - Εκκρεμής κριτικές - 2

Αφού ο κριτής έχει απορρίψει την έρευνα δίνοντας πιθανές διορθώσεις στο χρήστη, ο χρήστης καταθέτει τη διορθωμένη έρευνα και η διαδικασία κριτικής ξεκινάει ξανά από την αρχή.

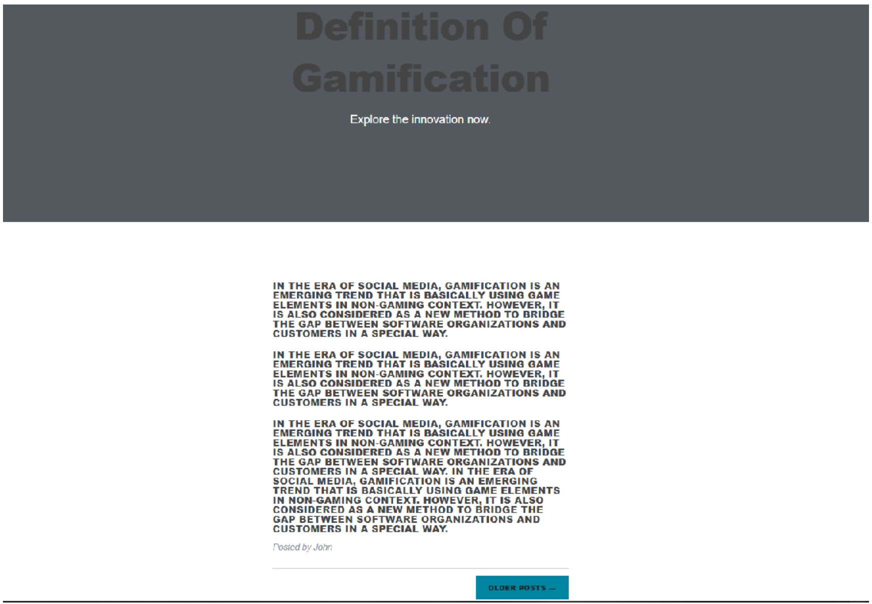
Όταν ο κριτής αποφασίσει ότι η έρευνα είναι ικανή να παρουσιαστεί στο συνέδριο, πατώντας το κουμπί με την ένδειξη ‘Approve’, η κατάσταση την κριτικής τίθεται σε ‘Approved’ και η έρευνα είναι έτοιμη να παρουσιαστεί στο συνέδριο.

ΥΠΟΚΕΦΑΛΑΙΟ 4.7 – ΛΟΙΠΕΣ ΛΕΙΤΟΥΡΓΙΕΣ

Στα προηγούμενα υποκεφάλαια είδαμε τις βασικές και κύριες λειτουργίες της ιστοσελίδας για τη διαχείριση αυτής αλλά και τη διαδικασία κριτικής. Σε αυτό το κεφάλαιο θα αναλυθούν οι διάφορες άλλες λειτουργίες της ιστοσελίδας και το υλικό που παρουσιάζεται σε αυτήν.

- <u>Home page:</u>
  Η αρχική σελίδα της ιστοσελίδας, όπως έχουμε ήδη προαναφέρει, περιλαμβάνει άρθρα τα οποία έχουν δημοσιευθεί στην ιστοσελίδα. Αυτά, τα συνοδεύουν αντίστοιχες εικόνες, καθώς και ο τίτλος του άρθρου. Οι μορφές των άρθρων παρουσιάζονται μέσω του carousel είτε σε λίστα κάτω από το carousel.

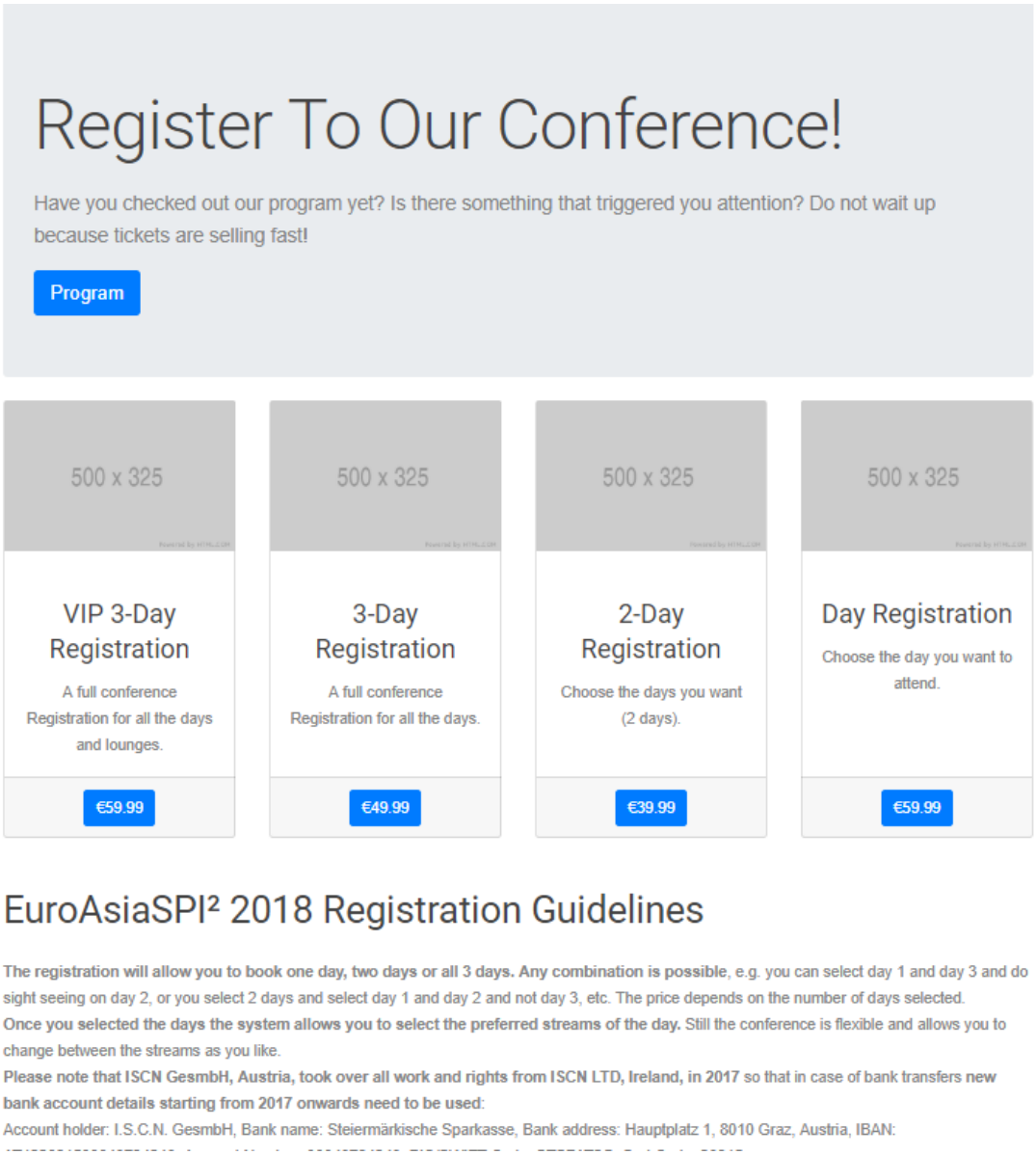
- <u>News και Post page:</u>
  Στην σελίδα ‘News’ μπορούμε να δούμε τη λίστα των δημοσιευμένων άρθρων. Επισκέπτοντας ένα άρθρο, μεταφερόμαστε στη σελίδα όπου και μπορούμε να διαβάσουμε ολόκληρο το άρθρο.



Εικόνα 24 - Προβολή άρθρου

- Tickets page:

  Στη σελίδα 'Tickets' ο χρήστης μπορεί να πληροφορηθεί για το κόστος συμμετοχής στο συνέδριο, τα διαθέσιμα εισιτήρια και τις τιμές του, τα οποία ρυθμίζονται από το διαχειριστή του συστήματος.



Εικόνα 25 - Σελίδα εισιτηρίων

- Submit a paper page:

  Ο χρήστης μπορεί να βρει χρήσιμες πληροφορίες για τη διαδικασία συμμετοχής στο συνέδριο. Η σελίδα αυτή περιλαμβάνει οδηγίες πως να καταθέσεις μια έρευνα ώστε να παρουσιαστεί στο συνέδριο, κ.α. Το υλικό αυτής το έχει θέσει ο διαχειριστής του συστήματος.

- About page:

  Η σελίδα 'About' περιέχει κείμενο το οποίο έχει ως βασικό στόχο την ενημέρωση του χρήστη για το τι ακριβώς συμπεριλαμβάνεται σε αυτό το συνέδριο, πληροφορίες που αφορούν τη διοργάνωση, το μέρος, ιστορικό αυτού κ.α.

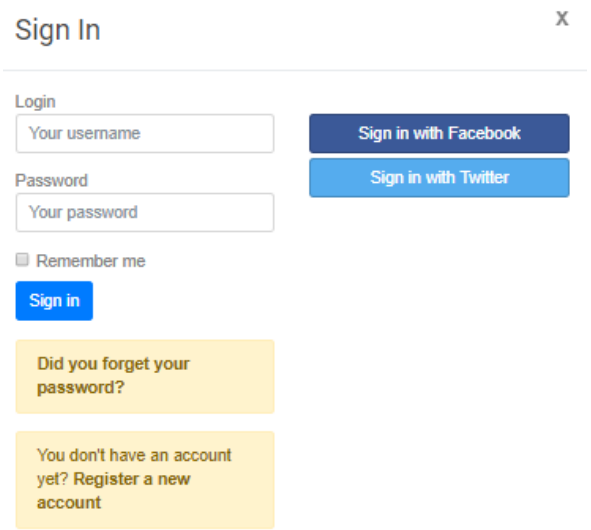
- Program Page:

  Στη σελίδα αυτή, όπως ήδη έχουμε αναλύσει, παρουσιάζεται το πρόγραμμα του συνεδρίου σε μορφή timeline. Συμπεριλαμβάνεται ο τίτλος της παρουσίασης, η ώρα και άλλες χρήσιμες πληροφορίες.

- Register/Login/Forgot Password:

  Κάτω από την επιλογή 'Register' έχουμε τη δυνατότητα να κάνουμε εγγραφή στην ιστοσελίδα, επιλέγοντας τον ρόλο μας, είτε είναι απλός χρήστης, είτε είναι κριτής.

  Στο popup που εμφανίζεται κατά τη διαδικασία εισόδου στην ιστοσελίδα, μας δίνεται η δυνατότητα να επιλέξουμε να κάνουμε επαναφορά κωδικού, σε περίπτωση μη δυνατής πρόσβασης στο λογαριασμό μας.



Εικόνα 26 - Φόρμα εισόδου

- Footer:

  Το footer της ιστοσελίδας εμφανίζεται σε κάθε σελίδα. Αυτό συμπεριλαμβάνει γενικές συνοπτικές πληροφορίες για το συνέδριο, όπως είναι η διεύθυνση της έδρας του συνεδρίου, τα social media προφιλ της εταιρείας, τηλέφωνα επικοινωνίας κ.λπ.

Εικόνα 27 - Footer

ΚΕΦΑΛΑΙΟ 5 – ΣΥΜΠΕΡΑΣΜΑΤΑ

Η υπηρεσία που αναπτύχθηκε δίνει την δυνατότητα στους χρήστες να εισέρχονται γρήγορα και με ευκολία στο σύστημα. Οι χρήστες μπορούν να βλέπουν απευθείας τα επερχόμενα προγράμματα συνεδρίων και την κατάσταση των δημοσιεύσεων που έχουν υποβάλλει.

Επιπλέον η καινοτομία που προσφέρει η υπηρεσία είναι ότι ο χρήστης μπορεί να βλέπει όλες τις πολλαπλές υποβολές που έχει κάνει σε μια σελίδα αλλά και να παρακολουθεί την κατάσταση τους. Ο διαχειριστής της εφαρμογής μπορεί να έχει πρόσβαση στις δημοσιεύσεις των χρηστών οποιαδήποτε χρονική στιγμή και από οποιαδήποτε συσκευή εκείνος θελήσει, αρκεί μόνο να συνδεθεί στο σύστημα. Αναθέτει τις έρευνες που έχουν καταθέσει οι χρήστες στους κριτές, οι οποίοι θα αποφασίσουν μέσω της διαδικασίας του Review, αν η έρευνα είναι ικανή για παρουσίαση στο συνέδριο.

Τέλος επειδή χρησιμοποιείται η αρχιτεκτονική σχεδίασης MVC μπορούν άμεσα να εισαχθούν και να αλληλεπιδράσουν νέες οντότητες και λειτουργίες με το σύστημα. Η εφαρμογή με λίγες τροποποιήσεις μπορεί να χρησιμοποιηθεί όχι μόνο για την υποβολή δημοσιεύσεων αλλά σαν μία ολοκληρωμένη πλατφόρμα για συνέδρια που θα περιέχει ανακοινώσεις, υλικό για ενημέρωση χρηστών κ.α. η οποία είναι εξ’ ολοκλήρου παραμετροποιήσιμη από το διαχειριστή του συστήματος.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] *Η ΙΣΤΟΡΙΑ ΤΟΥ INTERNET* (December 2018)

http://www.uth.gr/main/help/help-desk/internet/internet3.html

[2] Σπυρίδων Δ. Παχώμης (2017). *Το πλαίσιο λογισμικού Angular 5 για την ανάπτυξη εφαρμογών παγκόσμιου ιστού*

[3] Αναστάσιος Γ. Σταθόπουλος (2009). *Ανάπτυξη Εφαρμογών Διαχείρισης Διαδικασιών σε Περιβάλλον Java Spring*

[4] Καριωτάκη Σοφία (2013). *Ανάπτυξη δυναμικής διαδικτυακής εφαρμογής για κοινότητα συγγραφέων και αναγνωστών*

[5] Δρ. Μιχάλης Σαλαμπάσης (2008), *Εισαγωγή στον προγραμματισμό διαδικτυακών εφαρμογών*

[6] Στερεή- Αναστασία Γ. Καρούσου (2017). *Ανάπτυξη Portal με Spring για Διαχείριση Άυλων Υποβολών.*

[7] Duckett, J. (2011). *HTML and CSS: design and build websites. Indianopolis: Wiley & Sons*.

[8] Frain, B. (2015). *Responsive web design with HTML5 and CSS3*. *Birmingham, UK: Packt Publishing Limited.*

[9] Kamil Kolonko (2018). *Performance comparison of the most popular relational and non-relational database management systems*

[10] Marcus, A. (2013). *Design, user experience, and usability. Berlin: Springer*.

[11] Marijn Haverbeke (2018), *Eloquent JavaScript*

[12] Stanislav Nazmutdinov (2015). *Web application front end architecture and development using AngularJS framework*

[13] Tu Nguyen (2018). *Java Spring Framework in developing the Knowledge Article Management application*

[14] *Angular Documentation* (October 2018)

https://angular.io/docs

[15] *Angular 4 Java Developers* (October 2018)

https://www.udemy.com/angular-4-java-developers/learn/v4/overview

[16] *Angular 7 (formerly Angular 2) - The Complete Guide* (November 2018)

https://www.udemy.com/the-complete-guide-to-angular-2/learn/v4/overview

[17] *Bootstrap Documentation* (October 2018)

https://getbootstrap.com/docs/4.1/getting-started/introduction/

[18] *Css styles, javascripts, php function examples* (October 2018)

http://www.tizag.comhttp://www.phpguru.org

http://www.developphp.com

http://www.phpfreaks.comhttp://www.java2s.com

[19] Encyclopedia, Britannica Concise. (October 2018) *HTML*

[20] Spring Documentation

https://spring.io/docs

[21] Spring Framework Master Class

https://www.udemy.com/spring-tutorial-for-beginners/learn/v4/overview

[22] https://stackoverflow.com

[23] Websystiqueadmin, “Spring 4 MVC+Hibernate 4+MySQL+Maven integration example using annotations”, Online Tutorial, (August 2014) http://websystique.com/springmvc/spring-4-mvc-andhibernate4-integration-example-using-annotations

http://www.answers.com/topic/html

ΠΑΡΑΡΤΗΜΑ – ΠΑΡΑΔΕΙΓΜΑ ΚΩΔΙΚΑ

Παράδειγμα κώδικα σχετικά με τα reviews του συστήματος. Στο ίδιο μοτίβο δημιουργήθηκαν και οι υπόλοιποι πίνακες, services, implementations και ο κώδικας του front-end.

Review.java

```
@Entity
@Table(name = "review")
@Cache(usage = CacheConcurrencyStrategy.NONSTRICT_READ_WRITE)
public class Review implements Serializable {

   private static final long serialVersionUID = 1L;

   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   private Long id;

   @Column(name = "status")
   private Boolean status;

   @Column(name = "notes")
   private String notes;

   @ManyToOne
   private Papers papers;

   @ManyToOne(optional = false)
   @NotNull
   private User assignee;

   @ManyToOne
   private User reviewer;
```

```
public Long getId() {
    return id;
}

public void setId(Long id) {
    this.id = id;
}

public Boolean isStatus() {
    return status;
}

public Review status(Boolean status) {
    this.status = status;
    return this;
}

public void setStatus(Boolean status) {
    this.status = status;
}

public String getNotes() {
    return notes;
}

public Review notes(String notes) {
    this.notes = notes;
    return this;
}

public void setNotes(String notes) {
    this.notes = notes;
}
```

```
public Papers getPapers() {
    return papers;
}

public Review papers(Papers papers) {
    this.papers = papers;
    return this;
}

public void setPapers(Papers papers) {
    this.papers = papers;
}

public User getAssignee() {
    return assignee;
}

public Review assignee(User user) {
    this.assignee = user;
    return this;
}

public void setAssignee(User user) {
    this.assignee = user;
}

public User getReviewer() {
    return reviewer;
}

public Review reviewer(User user) {
    this.reviewer = user;
    return this;
```

```
}

public void setReviewer(User user) {
    this.reviewer = user;
}

@Override
public boolean equals(Object o) {
    if (this == o) {
        return true;
    }
    if (o == null || getClass() != o.getClass()) {
        return false;
    }
    Review review = (Review) o;
    if (review.getId() == null || getId() == null) {
        return false;
    }
    return Objects.equals(getId(), review.getId());
}

@Override
public int hashCode() {
    return Objects.hashCode(getId());
}

@Override
public String toString() {
    return "Review{" +
        "id=" + getId() +
        ", status='" + isStatus() + '\'' +
        ", notes='" + getNotes() + '\'' +
        '}';
```

```
    }
}
```

ReviewService.java

```
public interface ReviewService {

    /**
     * Save a review.
     *
     * @param reviewDTO the entity to save
     * @return the persisted entity
     */
    ReviewDTO save(ReviewDTO reviewDTO);

    /**
     *  Get all the reviews.
     *
     *  @param pageable the pagination information
     *  @return the list of entities
     */
    Page<ReviewDTO> findAll(Pageable pageable);

    /**
     *  Get the "id" review.
     *
     *  @param id the id of the entity
     *  @return the entity
     */
    ReviewDTO findOne(Long id);

    /**
```

```
    * Get the "id" review.
    *
    * @param id the id of the entity
    * @return the entity
    */
    List<ReviewDTO> findForUser();

    /**
    * Get the "id" review.
    *
    * @param id the id of the entity
    * @return the entity
    */
    List<ReviewDTO> findForSubmitter();

    /**
    * Delete the "id" review.
    *
    * @param id the id of the entity
    */
    void delete(Long id);
}
```

ReviewServiceImpl.java

```
package com.rfb.service.impl;

import com.rfb.service.ReviewService;
import com.rfb.domain.Review;
import com.rfb.repository.ReviewRepository;
import com.rfb.service.dto.ReviewDTO;
import com.rfb.service.dto.UserDTO;
import com.rfb.service.mapper.ReviewMapper;
import org.slf4j.Logger;
```

```
import org.slf4j.LoggerFactory;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import java.util.List;



/**
 * Service Implementation for managing Review.
 */
@Service
@Transactional
public class ReviewServiceImpl implements ReviewService{

    private final Logger log = LoggerFactory.getLogger(ReviewServiceImpl.class);

    private final ReviewRepository reviewRepository;

    private final ReviewMapper reviewMapper;

    public ReviewServiceImpl(ReviewRepository reviewRepository, ReviewMapper
reviewMapper) {
        this.reviewRepository = reviewRepository;
        this.reviewMapper = reviewMapper;
    }

    /**
     * Save a review.
     *
     * @param reviewDTO the entity to save
     * @return the persisted entity
     */
```

```
@Override
public ReviewDTO save(ReviewDTO reviewDTO) {
    log.debug("Request to save Review : {}", reviewDTO);
    Review review = reviewMapper.toEntity(reviewDTO);
    review = reviewRepository.save(review);
    return reviewMapper.toDto(review);
}

/**
 *  Get all the reviews.
 *
 *  @param pageable the pagination information
 *  @return the list of entities
 */
@Override
@Transactional(readOnly = true)
public Page<ReviewDTO> findAll(Pageable pageable) {
    log.debug("Request to get all Reviews");
    return reviewRepository.findAll(pageable)
        .map(reviewMapper::toDto);
}

/**
 *  Get one review by id.
 *
 *  @param id the id of the entity
 *  @return the entity
 */
@Override
@Transactional(readOnly = true)
public ReviewDTO findOne(Long id) {
    log.debug("Request to get Review : {}", id);
    Review review = reviewRepository.findOne(id);
    return reviewMapper.toDto(review);
```

```
    }

    @Override
    @Transactional(readOnly = true)
    public List<ReviewDTO> findForUser() {
        List<Review> review = reviewRepository.findByReviewerIsCurrentUser();
        return reviewMapper.toDto(review);
    }

    @Override
    @Transactional(readOnly = true)
    public List<ReviewDTO> findForSubmitter() {
        List<Review> review = reviewRepository.findByAssigneeIsCurrentUser();
        return reviewMapper.toDto(review);
    }


    /**
     *  Delete the  review by id.
     *
     *  @param id the id of the entity
     */
    @Override
    public void delete(Long id) {
        log.debug("Request to delete Review : {}", id);
        reviewRepository.delete(id);
    }
}
```

ReviewModel.ts

```
import { BaseEntity } from './../../shared';

export class Review implements BaseEntity {
```

```
    constructor(
        public id?: number,
        public status?: boolean,
        public notes?: string,
        public papersId?: number,
        public papersBody?: string,
        public assigneeId?: number,
        public papersStatus?: string,
        public papersAuthors?: string,
        public reviewerId?: number,
    ) {
        this.status = false;
    }
}
```

reviewer.component.ts

```
import {Component, Input, OnInit} from '@angular/core';
import {ActivatedRoute, ActivatedRouteSnapshot, NavigationEnd, Router} from
'@angular/router';
import {Title} from '@angular/platform-browser';
import {Review, ReviewService} from '../../entities/review';
import {Principal, ResponseWrapper, User, UserService} from '../../shared';
import {JhiAlertService, JhiEventManager, JhiParseLinks} from 'ng-jhipster';
import {Observable, Subscription} from 'rxjs';
import {Papers, PapersService} from '../../entities/papers';
import {Response} from '@angular/http';

@Component({
    selector: 'jhi-reviewer',
    templateUrl: './reviewer.component.html',
```

```
    styleUrls: [
        'reviewer.css'
    ]
})
export class ReviewerComponent implements OnInit {

    public currentRoute: string;
    currentAccount: any;
    reviews: Review[];
    personalReviews: any;
    error: any;
    success: any;
    links: any;
    totalItems: any;
    queryCount: any;
    eventSubscriber: Subscription;
    itemsPerPage: any;
    page: any;
    predicate: any;
    reverse: any;
    indicator: any;
    notif: any;
    file: string;
    isSaving: boolean;
    myPendingPapers: any;
    allPapers: any;
    reviewerEmail: any;
    submitterEmail: any;
    updatePaper: Papers;
    paperIdNumber: any;
    unassignedPapers: any;
    reviewers: any;
    newReview: any;
```

```
    constructor(private titleService: Title,
                private router: Router,
                private reviewService: ReviewService,
                private eventManager: JhiEventManager,
                private principal: Principal,
                private activatedRoute: ActivatedRoute,
                private parseLinks: JhiParseLinks,
                private alertService: JhiAlertService,
                private papersService: PapersService,
                private userService: UserService) {}

    private getPageTitle(routeSnapshot: ActivatedRouteSnapshot) {
        let title: string = (routeSnapshot.data && routeSnapshot.data['pageTitle']) ?
routeSnapshot.data['pageTitle'] : 'Reviewer';
        if (routeSnapshot.firstChild) {
            title = this.getPageTitle(routeSnapshot.firstChild) || title;
        }
        this.currentRoute = this.router.url;
        return title;
    }
    ngOnInit() {
        this.loadAll();
        this.principal.identity().then((account) => {
            this.currentAccount = account;
        });

        this.router.events.subscribe((event) => {
            if (event instanceof NavigationEnd) {

this.titleService.setTitle(this.getPageTitle(this.router.routerState.snapshot.root));
            }
        });
        this.registerChangeInReviews();
        this.registerChangesInPapers();
```

```
        this.registerChangeInUser();

    }
    //use this to upload instantly to the object
    onFileName(fileName: string) {
        this.file = fileName;
    }

    setNotes(review) {
        review.notes = this.file;
        this.reviewService.update(review).subscribe(
            (response) => {
                if (response == 200) {
                    this.error = null;
                    this.success = 'OK';
                    this.loadAll();
                } else {
                    this.success = null;
                    this.error = 'ERROR';
                }
            });
    }

    loadAll() {
        if (this.principal.hasAnyAuthorityDirect(['ROLE_ADMIN'])) {
            this.reviewService.query().subscribe((res: ResponseWrapper) => {
                this.reviews = res.json;
            }, (res: ResponseWrapper) => this.onError(res.json));
        }
        if (this.principal.hasAnyAuthorityDirect(['ROLE_ORGANIZER',
'ROLE_ADMIN'])) {
            this.reviewService.findForUser()
                .subscribe((res: ResponseWrapper) => {
                    this.personalReviews = res.json;
```

```
            }, (res: ResponseWrapper) => this.onError(res.json));
        }
        if (this.principal.hasAnyAuthorityDirect(['ROLE_USER'])) {
            this.reviewService.findForSubmitter()
                .subscribe((res: ResponseWrapper) => {
                    this.personalReviews = res.json;
                }, (res: ResponseWrapper) => this.onError(res.json));
            this.reviewService.findPapersForUser()
                .subscribe((res: ResponseWrapper) => {
                    this.myPendingPapers = res.json;
                }, (res: ResponseWrapper) => this.onError(res.json));
        }
        if (this.principal.hasAnyAuthorityDirect(['ROLE_ADMIN'])) {
            this.papersService.query()
                .subscribe((res: ResponseWrapper) => {
                    this.allPapers = res.json;
                }, (res: ResponseWrapper) => this.onError(res.json));
        }
        if (this.principal.hasAnyAuthorityDirect(['ROLE_ADMIN'])) {
            this.papersService.finUnassigned()
                .subscribe((res: ResponseWrapper) => {
                    this.unassignedPapers = res.json;
                }, (res: ResponseWrapper) => this.onError(res.json));
        }
        this.userService.findByAuthority('ROLE_ORGANIZER')
            .subscribe((res: ResponseWrapper) => { this.reviewers = res.json; }, (res:
ResponseWrapper) => this.onError(res.json));
    }

    notifyRev(user) {
        this.userService.find(user).subscribe((user) => {
                this.reviewerEmail = user.email;
            this.userService.notifyReviewer(this.reviewerEmail).subscribe(() => {
                this.error = null;
```

```
                this.success = 'OK';
            }, () => {
                this.success = null;
                this.error = 'ERROR';
            });
        });
    }

    notifySubmitter(user) {
        this.userService.find(user).subscribe((user) => {
            this.submitterEmail = user.email;
            this.userService.notifySubmitter(this.submitterEmail).subscribe(() => {
                this.error = null;
                this.success = 'OK';
            }, () => {
                this.success = null;
                this.error = 'ERROR';
            });
        });
    }

    registerChangesInPapers() {
        this.eventSubscriber = this.eventManager.subscribe('papersListModification',
(response) => this.loadAll());
    }

    sort() {
        const result = [this.predicate + ',' + (this.reverse ? 'asc' : 'desc')];
        if (this.predicate !== 'id') {
            result.push('id');
        }
        return result;
    }
```

```
setApproved(paper, isApproved) {
    paper.status = isApproved;

    this.reviewService.update(paper).subscribe(
        (response) => {
            if (response == 200) {
                this.error = null;
                this.success = 'OK';
                this.loadAll();
            } else {
                this.success = null;
                this.error = 'ERROR';
            }
        });
    if (isApproved) {
        this.setStatus(paper, "Approved");
    } else {
        this.setStatus(paper, "Rejected");
    }
}

accept(review, isApproved){
    if (isApproved) {
        review.status = isApproved;
        this.reviewService.update(review).subscribe(
            (response) => {
                if (response == 200) {
                    this.error = null;
                    this.success = 'OK';
                    this.loadAll();
                } else {
                    this.success = null;
                    this.error = 'ERROR';
                }
```

```
            });
        this.setStatus(review, "Ready For Review");
    } else {
        this.reviewService.delete(review.id).subscribe((response) => {
            this.eventManager.broadcast({
                name: 'reviewListModification',
                content: 'Deleted an review'
            });
        });
        this.setStatus(review, null);
        this.loadAll();
    }
}

setStatus(paper, status) {
    this.paperIdNumber = paper.papersId;

    this.papersService.find(this.paperIdNumber).subscribe((papers) => {
        this.updatePaper = papers;
        this.updatePaper.status = status;
        this.subscribeToSaveResponse(
            this.papersService.update(this.updatePaper));
        if (status == 'Ready For Review') {
            this.notifyRev(paper.reviewerLogin);
        }
        if (status == 'Review Failed') {
            this.notifySubmitter(paper.assigneeLogin);
        }
    });
}

assignToReview(review, selectedReviewer) {
    this.newReview = Review;
    this.newReview.assigneeId = null;
```

```
        this.newReview.assigneeId = review.userId;
        this.newReview.papersId = review.id;
        this.newReview.papersBody = review.body;
        this.newReview.reviewerId = selectedReviewer;
        this.subscribeToSaveResponseOfReview(
            this.reviewService.create(this.newReview));
            }

    private subscribeToSaveResponseOfReview(result: Observable<Review>) {
        result.subscribe((res: Review) =>
            this.onSaveSuccessReview(res), (res: Response) => this.onSaveError());
    }

    private subscribeToSaveResponse(result: Observable<Papers>) {
        result.subscribe((res: Papers) =>
            this.onSaveSuccess(res), (res: Response) => this.onSaveError());
    }
    private onSaveSuccess(result: Papers) {
        this.eventManager.broadcast({ name: 'papersListModification', content:
'OK'});
        this.isSaving = false;
    }

    private onSaveSuccessReview(result: Review) {
        this.eventManager.broadcast({ name: 'reviewListModification', content:
'OK'});
        this.isSaving = false;
    }

    private onSaveError() {
        this.isSaving = false;
    }
    registerChangeInReviews() {
        this.eventManager.subscribe('reviewListModification', (response) =>
```

```
this.loadAll());
    }

    registerChangeInUser() {
        this.eventManager.subscribe('userListModification', (response) =>
this.loadAll());
    }

    private onSuccess(data, headers) {
        this.links = this.parseLinks.parse(headers.get('link'));
        this.totalItems = headers.get('X-Total-Count');
        this.queryCount = this.totalItems;
        this.reviews = data;
    }

    private onError(error) {
        this.alertService.error(error.error, error.message, null);
    }

}
```

reviewer.component.html

```
<div *jhiHasAnyAuthority="'ROLE_USER'">
    <h2 >
        <span>My Submitted Papers</span>
    </h2>
    <button class="btn btn-primary float-right jh-create-entity create-review"
[routerLink]="['/', { outlets: { popup: ['papers-new'] } }]">
    <span class="fa fa-plus"></span>
    <span >
```

```
            Submit your paper
            </span>
    </button>
    <div class="row">
    </div>
    <br/>
    <div class="table-responsive" *ngIf="myPendingPapers">
        <table class="table table-striped">
            <thead>
            <tr>
                <th ><span>Paper Id</span></th>
                <th ><span>Paper Name</span></th>
                <th ><span>Paper File</span></th>
                <th></th>
            </tr>
            </thead>
            <tbody>
            <tr *ngFor="let papers of myPendingPapers ;trackBy: trackId">
                <td>
                    <div *ngIf="papers.id">
                        <a [routerLink]="['../papers', papers.id  ]" >{{papers.id}}</a>
                    </div>
                </td>
                <td>
                    {{papers.name}}
                </td>
                <td><jhi-details-upload [fileUpload]='papers.body'>{{papers.body}}</jhi-
details-upload></td>

                <td class="text-right">
                    <button type="submit"
                            [routerLink]="['/', { outlets: { popup: 'papers/'+ papers.id + '/edit'}
}]"
                            replaceUrl="true"
```

```
                    class="btn btn-primary btn-sm"
                    *jhiHasAnyAuthority="['ROLE_USER']">
                <span class="fa fa-pencil"></span>
                <span class="d-none d-md-inline">Edit</span>
            </button>
            <button type="submit"
                    [routerLink]="['/', { outlets: { popup: 'papers/'+ papers.id +
'/delete'} }]"
                    replaceUrl="true"
                    class="btn btn-danger btn-sm"
                    *jhiHasAnyAuthority="['ROLE_USER']">
                <span class="fa fa-remove"></span>
                <span class="d-none d-md-inline">Delete</span>
            </button>
        </td>
    </tr>
    </tbody>
</table>
</div>
</div>
<hr>


<div>
    <h2>
        <span>Pending Reviews</span>
    </h2>
    <h2 *jhiHasAnyAuthority="['ROLE_ADMIN']">
        <button class="btn btn-primary float-right jh-create-entity create-review"
[routerLink]="['/', { outlets: { popup: ['review-new'] } }]">
            <span class="fa fa-plus"></span>
            <span >
            Create new Review
            </span>
```

```
        </button>
    </h2>
    <div class="row">
    </div>
    <br/>
    <div class="table-responsive" *ngIf="personalReviews">
        <table class="table table-striped">
            <thead>
            <tr>
                <th ><span>Paper Title</span></th>
                <th ><span>Authors</span></th>
                <th ><span>Paper</span></th>
                <th ><span>Username</span></th>
                <th ><span>Notes</span></th>
                <th ><span>Status</span></th>
                <th></th>
            </tr>
            </thead>
            <tbody>
            <tr *ngFor="let review of personalReviews ;trackBy: trackId">
                <td>
                    <div *ngIf="review.papersId">
                        <a [routerLink]="['../papers', review.papersId  ]"
>{{review.papersName}}</a>
                    </div>
                </td>
                <td>
                    {{review.papersAuthors}}
                </td>
                <!--This is to show the download link to the paper-->
                <td><jhi-details-upload
[fileUpload]='review.papersBody'>{{review.papersBody}}</jhi-details-upload></td>
                <td>
                    {{review.assigneeLogin}}
```

```
            </td>
            <!--this is to show the upload form and instantly save it to the review. It
also show the current file-->
            <td><jhi-form-upload *jhiHasAnyAuthority="['ROLE_ADMIN',
'ROLE_ORGANIZER']" [fileName]="review.notes"
(exportFileName)="onFileName($event)" (click)="setNotes(review,
review.notes)"></jhi-form-upload>
                <small *ngIf="review.notes!=null">Current File uploaded:
</small><jhi-details-upload [fileUpload]='review.notes'>{{review.notes}}</jhi-
details-upload></td>
            <td>{{review.papersStatus}}</td>
            <td class="text-right">
                <div class="btn-group flex-btn-group-container"
*jhiHasAnyAuthority="['ROLE_USER']">
                    <button class="btn btn-sm btn-success" (click)="setStatus(review,
'Ready For Review')" *ngIf="!review.status">Ready For Review</button>
                </div>
                <div *jhiHasAnyAuthority="['ROLE_USER']">
                    <small style="font-size: 11px;">Once ready, hit the button to notify
the reviewer. </small>
                </div>
                <div class="btn-group flex-btn-group-container"
*jhiHasAnyAuthority="['ROLE_ADMIN', 'ROLE_ORGANIZER']">
                    <button class="btn btn-success btn-sm" (click)="accept(review,
true)" *ngIf="review.status==null">Accept</button>
                    <button class="btn btn-danger btn-sm" (click)="accept(review,
false)" *ngIf="review.status==null">Reject</button>
                </div>

                <div class="btn-group flex-btn-group-container"
*jhiHasAnyAuthority="['ROLE_ADMIN', 'ROLE_ORGANIZER']">
                    <div *ngIf="review.papersStatus!=null">

                    <button class="btn btn-danger btn-sm"
```

```
(click)="setApproved(review, false)" *ngIf="review.status">Reject</button>
                    <button class="btn btn-success btn-sm"
(click)="setApproved(review, true)" *ngIf="!review.status">Approve</button>
                    </div>
                </div>
                <div class="btn-group flex-btn-group-container"
*jhiHasAnyAuthority="['ROLE_ADMIN', 'ROLE_ORGANIZER']">
                    <div *ngIf="review.papersStatus!=null">

                    <button class="btn btn-sm btn-danger" (click)="setStatus(review,
'Review Failed')" *ngIf="!review.status">Review Failed</button>
                    </div>
                </div>
                <div *jhiHasAnyAuthority="['ROLE_ORGANIZER']">
                    <small style="font-size: 11px;">The author will be notified about
review status. </small>
                </div>

            </td>
            <td class="text-right" *jhiHasAnyAuthority="['ROLE_ADMIN',
'ROLE_ORGANIZER']">
                <div class="btn-group flex-btn-group-container"
*jhiHasAnyAuthority="['ROLE_ADMIN', 'ROLE_ORGANIZER']">
                    <button class="btn btn-success btn-sm"
(click)="notifySubmitter(review.assigneeLogin)" >Notify Submitter</button>
                </div>
            </td>
        </tr>
        </tbody>
    </table>
</div>
<h5 *jhiHasAnyAuthority="['ROLE_ADMIN', 'ROLE_ORGANIZER']">Please
follow the <a href="https://easychair.org/publications/easychair.docx">submission
guidelines</a> in order to comply with the correct format while uploading the
```

```
review notes.</h5>
    <h3 *ngIf="personalReviews==0"> No pending reviews!</h3>
</div>

<hr>



<div *jhiHasAnyAuthority="['ROLE_ADMIN']">
    <h2>
        <span>All Reviews</span>
    </h2>
    <div class="row">
    </div>
    <br/>
    <div class="table-responsive" *ngIf="reviews">
        <table class="table table-striped">
            <thead>
            <tr>
                <th ><span>Paper Title</span></th>
                <th ><span>Paper</span></th>
                <th ><span>Submitter's Username</span></th>
                <th ><span>Reviewer's Username</span></th>
                <th ><span>Notes</span></th>
                <th ><span>Status</span></th>
                <th></th>
            </tr>
            </thead>
            <tbody>
            <tr *ngFor="let review of reviews ;trackBy: trackId">
                <td>
                    <div *ngIf="review.papersId">
                        <a [routerLink]="['../papers', review.papersId  ]"
>{{review.papersName}}</a>
                    </div>
```

```
            </td>
            <!--This is to show the download link to the paper-->
            <td><jhi-details-upload
[fileUpload]='review.papersBody'>{{review.papersBody}}</jhi-details-upload></td>
            <td>
                {{review.assigneeLogin}}
            </td>
            <td>
                {{review.reviewerLogin}}
            </td>
            <!--this is to show the upload form and instantly save it to the review. It
also show the current file-->
            <td><small *ngIf="review.notes!=null">Current File uploaded:
</small><jhi-details-upload [fileUpload]='review.notes'>{{review.notes}}</jhi-
details-upload></td>
            <td>{{review.papersStatus}}</td>
            <td class="text-right">
                <div class="btn-group flex-btn-group-container"
*jhiHasAnyAuthority="['ROLE_ADMIN', 'ROLE_ORGANIZER']">
                    <button class="btn btn-success btn-sm"
(click)="notifyRev(review.reviewerLogin)" >Notify Reviewer</button>
                </div>
            </td>
            <td class="text-right">
                <div class="btn-group flex-btn-group-container"
*jhiHasAnyAuthority="['ROLE_ADMIN', 'ROLE_ORGANIZER']">
                    <button class="btn btn-success btn-sm"
(click)="notifySubmitter(review.assigneeLogin)" >Notify Submitter</button>
                </div>
            </td>
        </tr>
        </tbody>
    </table>
</div>
```

```
</div>

<hr>

<div *jhiHasAnyAuthority="['ROLE_ADMIN']">
    <h2>
        <span>All Papers</span>
    </h2>
    <div class="row">
    </div>
    <br/>
    <div class="table-responsive" *ngIf="allPapers">
        <table class="table table-striped">
            <thead>
            <tr>
                <th ><span>Paper Name</span></th>
                <th ><span>Paper</span></th>
                <th ><span>Authors</span></th>
                <th ><span>Username</span></th>
                <th></th>
            </tr>
            </thead>
            <tbody>
            <tr *ngFor="let review of allPapers ;trackBy: trackId">
                <td>
                    <div *ngIf="review.id">
                        <a [routerLink]="['../papers', review.id  ]" >{{review.name}}</a>
                    </div>
                </td>
                <!--This is to show the download link to the paper-->
                <td><jhi-details-upload [fileUpload]='review.body'>{{review.body}}</jhi-
details-upload></td>
                <td>
                    {{review.authors}}
```

```
            </td>
            <td>
                {{review.userLogin}}
            </td>
            <!--this is to show the upload form and instantly save it to the review. It
also show the current file-->
            <td class="text-right">
                <div class="btn-group flex-btn-group-container"
*jhiHasAnyAuthority="['ROLE_ADMIN']">
                </div>
            </td>
        </tr>
        </tbody>
    </table>
  </div>
</div>


<hr>

<div *jhiHasAnyAuthority="['ROLE_ADMIN']">
  <h2>
    <span>Unassigned Papers</span>
  </h2>
  <div class="row">
  </div>
  <br/>
  <div class="table-responsive" *ngIf="unassignedPapers">
    <table class="table table-striped">
      <thead>
      <tr>
        <th ><span>Paper Name</span></th>
        <th ><span>Paper</span></th>
        <th ><span>Authors</span></th>
```

```
            <th ><span>Username</span></th>
            <th></th>
        </tr>
        </thead>
        <tbody>
        <tr *ngFor="let review of unassignedPapers ;trackBy: trackId">
            <td>
                <div *ngIf="review.id">
                    <a [routerLink]="['../papers', review.id  ]" >{{review.name}}</a>
                </div>
            </td>
            <!--This is to show the download link to the paper-->
            <td><jhi-details-upload [fileUpload]='review.body'>{{review.body}}</jhi-details-upload></td>
            <td>
                {{review.authors}}
            </td>
            <td>
                {{review.userLogin}}
            </td>
            <!--this is to show the upload form and instantly save it to the review. It also show the current file-->
            <td class="text-right">
                <div class="btn-group flex-btn-group-container" *jhiHasAnyAuthority="['ROLE_ADMIN']">
                </div>
            </td>
            <td><select id="rev" name="rev" [(ngModel)]="selectedReviewer">
                <option [ngValue]="null"></option>
                <option [ngValue]="rev.id" *ngFor="let rev of reviewers">{{rev.login}}</option>
            </select>
                <button class="btn btn-success btn-sm" (click)="assignToReview(review, selectedReviewer)" >Assign</button></td>
```

```
            </tr>
            </tbody>
        </table>
    </div>
</div>
```