ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
& ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΠΛΑΤΦΟΡΜΑ ΔΙΑΧΕΙΡΙΣΗΣ ΑΙΤΗΜΑΤΩΝ ΒΛΑΒΩΝ ΚΤΗΡΙΩΝ ΖΕΠ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του/της
ΒΑΒΛΙΑΡΑΣ ΧΑΡΙΣΙΟΣ

Επιβλέπων: Καθηγητής Παντελής Αγγελίδης

ΚΟΖΑΝΗ/ΙΟΥΛΙΟΣ/2024

HELLENIC DEMOCRACY
UNIVERSITY OF WESTERN MACEDONIA
SCHOOL OF ENGINEERING
DEPARTMENT OF ELECTRICAL
& COMPUTER ENGINEERING
ece.uowm.gr

FAULT REQUEST MANAGEMENT PLATFORM FOR CAMPUS ZEP

THESIS

VAVLIARAS CHARISIOS

SUPERVISOR: Professor Pantelis Aggelidis

KOZANI/JULY/2024

ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
& ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΔΗΛΩΣΗ ΜΗ ΛΟΓΟΚΛΟΠΗΣ ΚΑΙ ΑΝΑΛΗΨΗΣ ΠΡΟΣΩΠΙΚΗΣ ΕΥΘΥΝΗΣ

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

Υπογραφή Φοιτητή:

Περίληψη

Στις μέρες μας η χρήση συστημάτων διαχείρισης βλαβών είναι αναγκαία καθώς οι προκλήσεις που αντιμετωπίζουν οι μεγάλες υποδομές ολοένα και αυξάνονται. Γενικότερα ένα σύστημα διαχείρισης βλαβών χρησιμοποιείται σε διάφορους οργανισμούς για να παρέχει λύσεις και υποστήριξη σε τεχνικά προβλήματα που προκύπτουν.

Αυτή η διπλωματική εργασία στοχεύει στην υλοποίηση ενός συστήματος διαχείρισης αιτημάτων βλαβών στα πλαίσια συντήρησης των κτηρίων ενός πανεπιστημίου. Μπορεί κανείς να συναντήσει υλοποιήσεις σε εφαρμογές κινητού (mobile apps) και διαδικτυακές εφαρμογές (web apps), για τη ανάπτυξη αυτής της εργασίας επιλέχθηκε η διαδικτυακή πλατφόρμα ανάπτυξης εφαρμογών APACHE, PHP, MySQL. Η επιλογή αυτή οφείλεται στο γεγονός ότι μια διαδικτυακή εφαρμογή είναι εύκολα προσβάσιμη και συμβατή σε κάθε συσκευή μέσα από ένα πρόγραμμα περιήγησης όπως για παράδειγμα το Google Chrome ή Firefox. Το αποτέλεσμα που προκύπτει απτήν υλοποίηση είναι μία διαδικτυακή εφαρμογή απλή και φιλική προς τον χρήστη.

Λέξεις Κλειδιά: Σύστημα διαχείρισης αιτημάτων βλαβών , διαδικτυακή πλατφόρμα, APACHE, PHP, MySQL, διαδικτυακή εφαρμογή.

Abstract

In today's world, the use of fault management systems is essential as the challenges faced by large infrastructures grow continuously. Generally, a fault management system is used in various organizations to provide solutions and support for technical issues that occur.

This thesis aims to implement a fault management system within the maintenance framework of a university's buildings. Implementations can be found in mobile applications (mobile apps) as well as web applications. For the development of this work, the web application development platform APACHE, PHP, MySQL was chosen. This choice is due to the fact that a web application is easily accessible and compatible on every device through a web browser such as Google Chrome or Firefox. The result of this implementation is a web application simple and user-friendly.

Keywords: fault management system, web platform, APACHE, PHP, MySQL, web application

Περιεχόμενα

Κατάλογος Εικόνων

Κεφάλαιο 1: Εισαγωγή

1.1 Η πρόκληση

Μεγάλη πρόκληση στην ευζωία ενός κτηρίου είναι ο τακτικός-έγκαιρος έλεγχος για τυχών βλάβες και προβλήματα και η άμεση αντιμετώπιση τους πρώτου αυτά υπερδιογκωθούν. Η δυσκολία αυτή οφείλεται σε διάφορους παράγοντες όπως : η φύση της κάθε βλάβης, η δυσκολία στον εντοπισμό της και το κόστος αντιμετώπισης. Καθοριστικό ρόλο επομένως παίζει η σωστή διαχείριση τους. Η διαχείριση βλαβών στα κτήρια ενός πανεπιστημίου δε, αποτελεί ζωτικό ρόλο στην λειτουργικότητα τους καθιστώντας τα βιώσιμα και αποδοτικά στην πάροδο του χρόνου.

1.2 Στόχος

Στόχος της εργασίας είναι να αντιμετωπίσει αυτή την πρόκληση με την δημιουργία ενός διαδικτυακού συστήματος που θα παρέχει άμεση επικοινωνία, σωστή οργάνωση και παρακολούθηση των αιτημάτων. Όραμα αυτής της εργασίας είναι να καταφέρει να καθιερωθεί ως μια νέα υπηρεσία του Πανεπιστημίου Δυτικής Μακεδονίας και να αποτελέσει σημαντικό εργαλείο στην όλη προσπάθεια διατήρησης και συντήρησης των κτηρίων του. Για να γίνει αυτό, είναι βασικό να γίνει εφικτή μια εύκολη καταγραφή βλαβών και στη συνέχεια η σωστή διαχείριση τους από την βάση δεδομένων στην οποία θα αποθηκεύονται. Με γνώμονα τα παραπάνω, η πλατφόρμα κατασκευάστηκε με τρόπο που την καθιστά απλή και φιλική προς τον χρήστη.

1.3 Σύντομη περιγραφή

Η λειτουργικότητα της πλατφόρμας που δημιουργήθηκε έχει ως εξής: Η πρόσβαση στην εφαρμογή διακρίνεται σε δύο επίπεδα, ο διαχειριστής (administrator) ο οποίος έχει πλήρη δικαιώματα στην πλατφόρμα και ο απλός χρήστης (Registered User) ο οποίος μπορεί να δηλώσει βλάβες στο σύστημα. Κάθε χρήστης αποκτά πρόσβαση δημιουργώντας ένα λογαριασμό, μέσω του διαδικτύου. Αφότου αποκτήσει πρόσβαση, έχει το δικαίωμα να υποβάλλει βλάβες που έχει εντοπίσει. Κάθε βλάβη πρέπει να συνοδεύεται από κάποιες σχετικές πληροφορίες όπως μια περιγραφή του προβλήματος, της σημαντικότητας του, σε ποιο τμήμα ανήκει κτλ. Η βλάβη, αφού πληροί τα παραπάνω, καταχωρείται στην βάση δεδομένων του συστήματος με την μορφή ticket. Από εκεί και μετά είναι προσβάσιμη στον διαχειριστή (admin), ο οποίος έχει την δυνατότητα να διαχειρίζεται πλήρως την πλατφόρμα. Αυτό σημαίνει πώς μπορεί να αναζητά, να φιλτράρει και να επεξεργάζεται το κάθε ticket ξεχωριστά, ακόμη μπορεί να δημιουργεί καινούριους λογαριασμούς χρηστών, να κάνει αλλαγές σε αυτούς ή να τους διαγράφει. Πολύ σημαντικό είναι επίσης το γεγονός ότι μπορεί να αλληλεπιδρά με τον χρήστη για την εξέλιξη κάθε βλάβης η ζητήματα που προκύπτουν, και αυτό γιατί, έτσι εξασφαλίζεται μια αποτελεσματική επικοινωνία μεταξύ τους. Τέλος αν η βλάβη έχει λυθεί ο διαχειριστής θέτει το ticket ως κλειστό για να ξεχωρίζει από τα μη επιλυμένα.

1.4 Εργαλεία που χρησιμοποιήθηκαν

Τα εργαλεία που χρησιμοποιήθηκαν για την ανάπτυξη αυτής της εφαρμογής αποτελούν: η γλώσσα σήμανσης HTML, τα Cascading Style Sheets (CSS), η MySQL για την διαχείριση της βάσης δεδομένων και οι γλώσσες PHP και JavaScript. Για την δημιουργία τοπικού server χρησιμοποιήθηκε το XAMPP Version 8.2.12, ένα πακέτο λογισμικού που περιλαμβάνει τον Apache Server και την MySQL. Επίσης χρησιμοποιήθηκε το ανοικτού κώδικα framework της Bootstrap και η βιβλιοθήκη jQuery. Ο editor που χρησιμοποιήθηκε για τον σκοπό αυτό είναι το Visual Studio Code.

Κεφάλαιο 2: Σχεδιασμός και ανάλυση Συστήματος

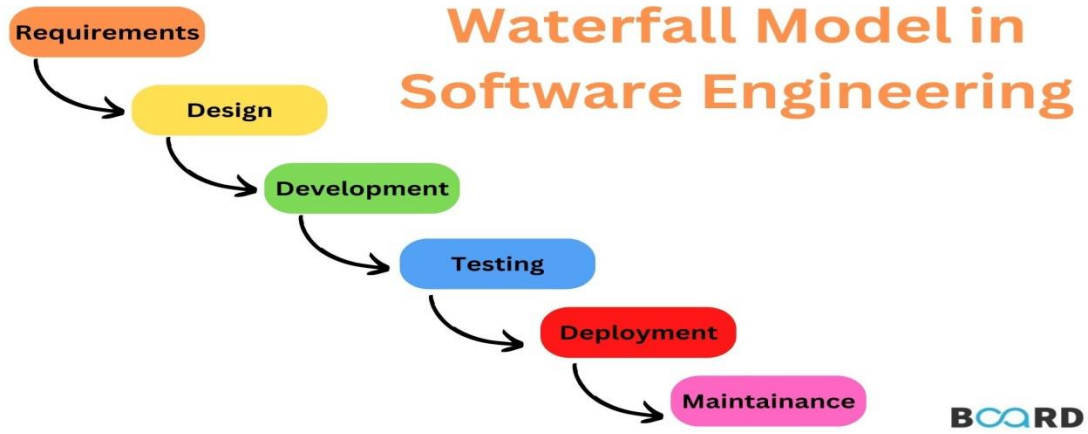
Κύριο μέλημα στην ανάπτυξη λογισμικού είναι ο σχεδιασμός και η ανάλυση των απαιτήσεων, και αυτό γιατί μόνο έτσι θα καταφέρουμε να διασφαλίσουμε τις κατάλληλες λειτουργίες και τα χαρακτηριστικά που πρέπει να περιλαμβάνονται στο τελικό προϊόν. Στο κεφάλαιο αυτό θα γίνει μια εκτενής περιγραφή του σχεδιασμού του συστήματος, τους χρήστες, του μοντέλου ανάπτυξης λογισμικού που χρησιμοποιήθηκε καθώς και τον ορολογιών που συναντά κανείς στην πλατφόρμα αυτή.

2.1 Μοντέλο ανάπτυξης λογισμικού

Στην βιομηχανία παραγωγής λογισμικού για το σχεδιασμό, την ανάπτυξη και την παραγωγή υψηλής ποιότητας και αξιόπιστου λογισμικού χρησιμοποιούνται διάφορα μοντέλα ανάπτυξης. Κάποια από αυτά είναι το μοντέλο καταρράκτη, το αυξητικό μοντέλο το επαναληπτικό μοντέλο και άλλα. Στην δική μας εργασία επιλέχθηκε το μοντέλο καταρράκτη, παρακάτω θα πούμε λίγα λόγια για αυτό.

Το μοντέλο καταρράκτη είναι το πρώτο μοντέλο ανάπτυξης λογισμικού που δημιουργήθηκε και αποτελεί βάση για τα επόμενα μοντέλα που ακολούθησαν. Το μοντέλο καταρράκτη αποτελείται από ξεχωριστές φάσεις καθορισμού απαιτήσεων και ανάπτυξης και ξεκινάνε μετά την ολοκλήρωση της προηγούμενης, και αυτό γιατί κάθε φάση βασίζεται σε πληροφορίες που συλλέγονται από την προηγούμενη. Τα πλεονεκτήματα του συγκεκριμένου μοντέλου σύμφωνα με τον S. Pargaonkar [1] είναι τα εξής: η διαδοχική φύση του κάνει τη διαδικασία ανάπτυξης προβλέψιμη και επιτρέπει σαφή προγραμματισμό εργασιών επίσης η λεπτομερής τεκμηρίωση που παρέχει, διασφαλίζει την πλήρη κατανόηση των απαιτήσεων του έργου, των προδιαγραφών σχεδιασμού και της υλοποίησης. Αρκετά σημαντικό επίσης το γεγονός ότι το μοντέλο καταρράκτη δίνει έμφαση στην παραγωγή απτών και καλά καθορισμένων παραδοτέων στο τέλος κάθε φάσης. Τέλος, είναι κατάλληλο για μικρά έργα με προκαθορισμένες απαιτήσεις. Αυτοί είναι και οι λόγοι που επιλέχθηκε το συγκεκριμένο

μοντέλο για την εργασία μας. Παρακάτω απεικονίζονται τα στάδια καθορισμού των απαιτήσεων.



Εικόνα 1: WATERFALL MODEL, ΙΣΤΟΣΕΛΙΔΑ https://bap-software.net

2.2 Ορολογίες / Χρήστες

2.2.1 Ορολογίες

1. **Tickets :** Είναι τα αιτήματα για την δήλωση μια βλάβης. Μαζί με την δημιουργία του ticket πρέπει να παρέχονται και κάποιες περεταίρω πληροφορίες οι οποίες είναι : α) Τύπος αιτήματος : Εδώ παρέχουμε μια πληροφορία σχετικά με τον τύπο της βλάβης, β) Μήνυμα: Εδώ παρέχουμε μια πιο λεπτομερή περιγραφή της βλάβης και είναι το μήνυμα που θα εμφανίζεται μέσα στην συνομιλία, γ) Κτήριο: Αναφερόμαστε στο κτήριο που αφορά η βλάβη.
2. **Χρήστες :** Είναι οι χρήστες της εφαρμογής. Διακρίνονται σε διαχειριστή και απλούς χρήστες. Για την δημιουργία λογαριασμού χρήστη απαιτείται ένα username ένα email και ο κωδικός.
3. **Κτήρια :** Προσδιορίζει το κτήριο στο οποίο εντοπίστηκε η βλάβη για την ευκολότερη αναζήτηση και επίλυση της κάθε αιτήματος βλάβης. Εισαγωγή κτηρίου έχει το δικαίωμα να δημιουργήσει μόνο ο διαχειριστής.
4. **Κατάσταση Ticket :** Το κάθε ticket μπορεί να οριστεί ως κλειστό η ανοικτό. Όταν είναι στην κατάσταση ανοικτό αυτό σημαίνει πως ακόμη δεν έχει επιλυθεί ενώ όταν είναι στην κατάσταση κλειστό μας δείχνει ότι έχει επιλυθεί και δεν χρειάζεται κάποια περαιτέρω ενέργεια. Την κατάσταση ενός ticket την ορίζει αποκλειστικά ο Διαχειριστής.
5. **Τύπος αιτήματος :** Σύντομη περιγραφή του τύπου της βλάβης για την καλύτερη κατηγοριοποίηση κάθε αιτήματος.

2.2.2 Χρήστες

1. **Διαχειριστής :** Είναι ο υπεύθυνος για την γενική διαχείριση του συστήματος και την επικοινωνία με όλους τους χρήστες. Ο διαχειριστής πρέπει να ελέγχει το κάθε αίτημα βλάβης και να αξιολογεί την εκάστοτε βλάβη ώστε να τις αναθέτει στο αρμόδιο τμήμα για επίλυση . Αυτή η διαδικασία ανάλογα με την σημαντικότητα της κάθε βλάβης πρέπει να γίνεται με σειρά προτεραιότητας, για αυτό ο διαχειριστής θα πρέπει να μπορεί να μπορεί να διακρίνει πότε μια βλάβη χρειάζεται άμεσα επιδιόρθωση η όχι.

2. **Κοινός χρήστης :** Ο ρόλος του είναι να υποβάλει τα αιτήματα βλαβών. Πρέπει να εξετάζει κάποιο αίτημα πριν το δηλώσει ώστε να μην φορτώνει το σύστημα με ανύπαρκτες η βλάβες, πρέπει επίσης να παρέχει σωστή περιγραφή της εκάστοτε βλάβης συμπεριλαμβάνοντας όλες τις σχετικές πληροφορίες.

2.3 Ανάλυση απαιτήσεων

Οι λειτουργικές απαιτήσεις του συστήματος μας είναι οι λειτουργίες και δυνατότητες που θα πρέπει να παρέχει το σύστημα μας ανάλογα με την ιδιότητα του κάθε χρήστη. Αυτές αναφέρονται παρακάτω όπως διαμορφώθηκαν σύμφωνα με τις ανάγκες μας.

1. **Ύπαρξη βάσης δεδομένων**

   • Αποθήκευσή πληροφοριών των αιτημάτων
   • Αποθήκευσή δεδομένων χρηστών

2. **Διαχείριση αιτημάτων βλαβών.**

   • Ο κοινός χρήστης μπορεί να δημιουργήσει αιτήματα βλαβών.
   • Ο κοινός χρήστης μπορεί να διαβάσει και να επεξεργαστεί/διαγράψει μονό τα αιτήματα βλάβης που δημιούργησε.
   • Ο διαχειριστής μπορεί να δημιουργήσει αιτήματα βλαβών.
   • Ο διαχειριστής μπορεί να επεξεργαστεί/διαγράψει το κάθε αίτημα που υπάρχει στην βάση δεδομένων.

3. **Διαχείριση χρήστη.**

   • Όλοι οι χρήστες κάνουν εγγραφή και είσοδο/έξοδο στην πλατφόρμα.
   • Ο κοινός χρήστης μπορεί να επεξεργαστεί μόνο τα προσωπικά του στοιχεία.
   • Ο διαχειριστής μπορεί να επεξεργαστεί τα στοιχεία κάθε λογαριασμού ή και να τον διαγράψει.
   • Ο διαχειριστής μπορεί να παραχωρήσει δικαιώματα διαχειριστή σε κάποιον άλλο λογαριασμό.
   • Ο διαχειριστής μπορεί να δημιουργήσει νέους λογαριασμούς χρηστών.

4. **Διαχείριση κτηρίων.**

• Μόνο ο διαχειριστής μπορεί να διαχειρίζεται την εισαγωγή και τη διαγραφή νέων κτηρίων.

Κεφάλαιο 3: Προγραμματισμός διαδικτύου- Υλοποίηση εφαρμογής

Η κατασκευή μιας διαδικτυακής εφαρμογής χωρίζεται σε δύο τμήματα τα οποία είναι γνωστά ως front-end και back-end. Ο όρος front-end αναφέρεται στο κομμάτι των γραφικών και της αλληλεπίδρασης με τον χρήστη ενώ ο όρος back-end σχετίζεται με το τμήμα της πλευράς του διακομιστή (server side). Γλώσσες όπως HTML, CSS και JavaScript ανήκουν στην πρώτη κατηγορία ενώ γλώσσες προγραμματισμού όπως PHP και SQL ανήκουν στην δεύτερη [2]. Όλα αυτά αποτελούν εργαλεία προγραμματισμού διαδικτύου και χρησιμεύουν στη κατασκευή ιστοτόπων, διαδικτυακών εφαρμογών και διαδικτυακών υπηρεσιών. Σε αυτό το κεφάλαιο θα μάθουμε πληροφορίες και θα δούμε τα χαρακτηριστικά για τα εργαλεία προγραμματισμού διαδικτύου που χρησιμοποιήθηκαν στην ανάπτυξη της εφαρμογής μας, επίσης θα πούμε μερικά λόγια για την υλοποίηση της εφαρμογής δίνοντας μερικά παραδείγματα για το πως χρησιμοποιήθηκε κάθε εργαλείο ξεχωριστά στην εφαρμογή μας. Στις πρώτες τρείς ενότητες θα αναφερθούμε στα front-end εργαλεία ενώ στις δύο τελευταίες στα back-end.

3.1 Η γλώσσα σήμανσης HTML

Εικόνα 2 : HTM5 LOGO, ΙΣΤΟΣΕΛΙΔΑ https://en.m.wikipedia.org

3.1.1 Η ιστορία της HTML

Η HTML (HyperText Markup Language) είναι μια τυπική γλώσσα σήμανσης για τη δημιουργία ιστοσελίδων. Η αρχική έκδοση της (HTML 1.0) κυκλοφόρησε για πρώτη φορά το 1993 με σκοπό την κοινή χρήση πληροφοριών μέσω του διαδικτύου, δημιουργός της ο

ερευνητής του CERN Tim Berners-Lee. Εκείνη την περίοδο δεν είχε μεγάλη απήχηση οπότε και δεν εξελίχθηκε ιδιαίτερα. Το 1995 έρχεται η HTML 2.0, προσθέτοντας νέες δυνατότητες στην αρχική έκδοση και βελτιώνοντας βασικά χαρακτηριστικά της, αυτή παραμένει η τυπική γλώσσα σήμανσης μέχρι και το 1997. Στη συνέχεια με την HTML 3.0. παρουσιάζεται ένα καινοτόμο προσχέδιο, το οποίο έδινε νέες δυνατότητες στην HTML και ισχυρά χαρακτηριστικά στο σχεδιασμό ιστοσελίδων. Δυστυχώς όμως δεν καθιερώθηκε, καθώς ο σχεδιασμός της επιβράδυνε αρκετά το πρόγραμμα περιήγησης .

Το 1998 καθιερώνεται η HTML 4.01 η οποία έφερε μεγάλη εξέλιξη στα πρότυπα HTML. Αποδείχθηκε πολύ επιτυχημένη έκδοση και για αυτό το λόγο χρησιμοποιήθηκε ευρέως για πολλά χρόνια. Τέλος το 2014 κυκλοφορεί η HTML 5.0 η οποία παραμένει μέχρι και σήμερα η πιο πρόσφατη έκδοση και χρησιμοποιείται παγκοσμίως από διάφορους browsers [3].

3.1.2 Η διάρθρωση της HTML

Η HTML αποτελεί την βάση του διαδικτύου καθώς ορίζει το περιεχόμενο και την δομή κάθε ιστότοπου. Είναι μια περιγραφική γλώσσα, χρησιμοποιείται δηλαδή για την περιγραφή του περιεχομένου μια σελίδας. Το πρώτο της αρχικό, συμβολίζει την λέξη “Hypertext” και εκφράζει το γεγονός ότι χρησιμοποιεί συνδέσμους που συνδέουν ιστοσελίδες μεταξύ τους δομώντας έτσι το διαδίκτυο. Η HTML αποτελείται από μια σειρά στοιχείων, τα οποία υποδεικνύουν πως θα εμφανίζεται το περιεχόμενο. Το όνομα ενός στοιχειού μέσα σε μια ετικέτα είναι ‘case-insensitive’ δηλαδή δεν έχει σημασία αν χρησιμοποιήσουμε πεζά ή κεφαλαία γράμματα στην γραφή [4].

Τα στοιχεία στην HTML έχουν την μορφή ετικετών (tags), υπάρχουν ετικέτες εκκίνησης (πχ.) ,ετικέτες τερματισμού (πχ.) αλλά και αυτόνομες ετικέτες οι οποίες ονομάζονται κενά στοιχεία (πχ.
). Ανάμεσα στις ετικέτες τερματισμού και εκκίνησης υπάρχει περιεχόμενο σε αντίθεση με τα κενά στοιχεία τα οποία είναι αυτόνομα, έχουν συγκεκριμένη συμπεριφορά και δεν ενσωματώνουν περιεχόμενο [5].
Ένα πρόγραμμα περιήγησης δεν εμφανίζει τον ακατέργαστο κώδικα που διαβάζει αλλά ερμηνεύει το περιεχόμενο της σελίδας ώστε να δημιουργήσει το λεγόμενο DOM (Document Object Model). Η διαδικασία αυτή ξεκινάει από πάνω προς τα κάτω διαβάζοντας το html έγγραφο, η σύνθεση του DOM γίνεται με την δημιουργία κόμβων κάθε φορά που ο browser συναντά ένα στοιχείο σήμανσης. Η δομή του DOM είναι μια αναπαράσταση της σελίδας σε μορφή δέντρου, οπού κάθε κόμβος αντιπροσωπεύει ένα στοιχείο html [6].

Ο κώδικας μιας τυπικής σελίδας HTML5 αντιπροσωπεύεται από την δομή DOM και αποτελείται από συγκεκριμένα στοιχεία τα οποία σύμφωνα με τον J. Krause [5] είναι:
<!DOCTYPE html> : Είναι μια πληροφορία για να αναγνωρίζει ο browser τον τύπο του εγγράφου που ακολουθεί. Είναι απαραίτητο κάθε html έγγραφο να ξεκινάει με αυτό.
<html> : Αποτελεί το root element του εγγράφου και περικλείει όλο το περιεχόμενο.

<body> : Μέσα σε αυτό γράφονται τα περιεχόμενα του εγγράφου, περιέχει δηλαδή το ορατό μερός της σελίδας.
<head> : Έχει πληροφορίες για το body.
<meta charset="utf-8"> : Είναι η κωδικοποίηση χαρακτήρων για το έγγραφο hmtl. Το "utf-8" είναι απο τα κύρια πρότυπα καθώς περιέχει όλους τους εδικούς χαρακτήρες.
<meta name="viewport"...> : Είναι υπεύθυνο για την προσαρμογή της προβολής του ιστότοπου ανάλογα με το μέγεθος της οθόνης
<title> : Ο τίτλος είναι το όνομα του αρχείου που εμφανίζεται στις καρτέλες του προγράμματος περιήγησης και για αυτό το λόγο πρέπει να δίνουμε τίτλους με νόημα.
<header> : Βρίσκεται πριν απτό body και συνήθως περιέχει ένα logo, έναν τίτλο και συνδέσμους πλοήγησης
<footer> : Είναι ένα σημασιολογικό στοιχείο που μπορεί να περιέχει τα πνευματικά δικαιώματα

template.html

```
1  <!DOCTYPE html>
2
3  <html lang="en">
4  <head>
5      <meta charset="utf-8">
6      <title>Page Title</title>
7  </head>
8
9  <body>
10
11 </body>
12 </html>
13
```

Εικόνα 3: A BASIC HTML5 PAGE, ΙΣΤΟΣΕΛΙΔΑ http://web.simmons.edu

3.1.3 Η χρήση της HTML στην εφαρμογή μας.

Η διεπαφή χρήστη της εφαρμογή μας σε κάθε σελίδα πλοήγησης που δημιουργήθηκε περιέχει HTML όποτε στα αρχεία tickets.php, departments.php, users.php, reply.php αλλά και σε όλες τι φόρμες του πρότζεκτ περιλαμβάνεται αρχικά η βασική δομή μιας HTML σελίδας και επιπρόσθετα ανάλογα με τις ανάγκες κάθε διεπαφής υπάρχουν διαφορετικά HTML στοιχεία όπως footer navbars ,tables κτλ. Στην εικόνα 4 για παράδειγμα φαίνεται ο πίνακας με τις πληροφορίες κάθε ticket ενώ στην εικόνα 5 η φόρμα εισαγωγής διαπιστευτηρίων για την είσοδο στην εφαρμογή.

```
103
104
105     <table id="ticketTable" class="table table-hover table-striped shadow ">
106         <tr>
107         <th>Ticket ID</th>
108         <th>Τύπος αιτήματος</th>
109         <th>Μήνυμα</th>
110         <th>Κτήριο</th>
111         <th>Κατάσταση Ticket</th>
112         <th>Δημιουργός-Χρήστης </th>
113         <th>Επεξεργασία Ticket</th>
114         <th>Ημερομηνία δημιουργίας</th>
115         <th> Συνομιλία</th>
116         </tr>
117
118
119
120     </table>
```

Εικόνα 4

```
41  <form action="" name="loginForm" onsubmit="return validateForm()" method="POST">
42
43
44
45    <div class="mb-3">
46  <label for = "username" class="form-label">Όνομα χρήστη</label>
47  <input type="text"  class="form-control" placeholder="Username" id="name" name="username"> </div>
48
49
50  <div class="mb-3">
51   <label for="password" class="form-label">Κωδικός πρόσβασης</label>
52  <input type="password" class="form-control" placeholder="Password" id ="pass" name="password">
53  </div>
54
55  <div class="mb-3">
56  <button type="submit" name="login_user" class="btn btn-primary">Είσοδος</button></div>
57
58  <div class="form-text" >Δημιουργία νέου λογαριασμού <a href="http://localhost/ticketing_system/forms/sign_up.php" class="link-underline-ligh
59
60  </form>
```

Εικόνα 5

3.2 CASCADING STYLE SHEETS

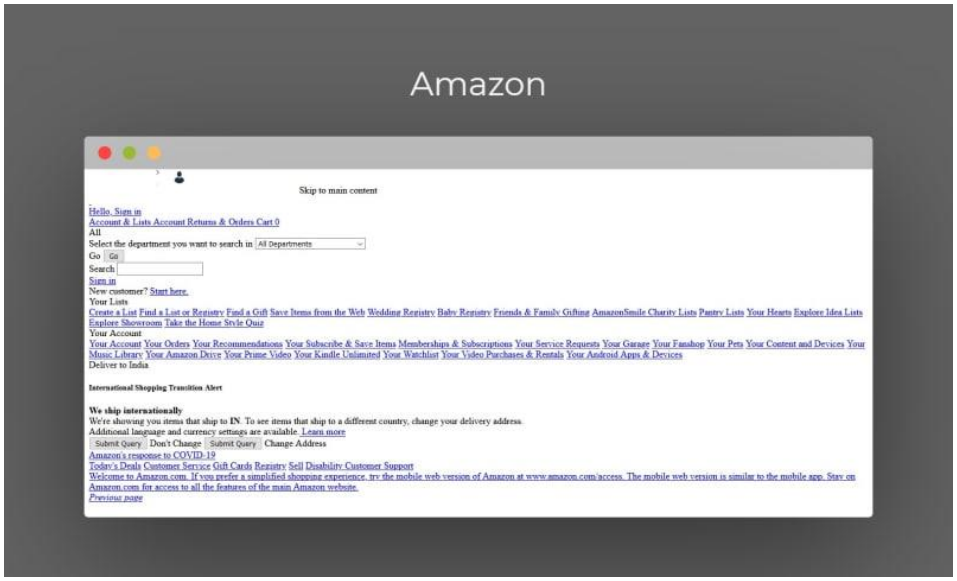
Εικόνα 6: CSS3 LOGO, ΙΣΤΟΣΕΛΙΔΑ https://en.wikipedia.org

3.2.1 Η ιστορία της CSS

Πίσω στο 1994, η html έχει ήδη κάνει την εμφάνιση της και το διαδίκτυο έχει αρχίσει να χρησιμοποιείται ως πλατφόρμα για ηλεκτρονικές εκδόσεις. Το πρόβλημα που υπήρχε ήταν πως δεν υπήρχε τρόπος για την διακόσμηση των εγγράφων. Ο Håkon Wium Lie εργαζόμενος στο CERN πάνω σε εξατομικευμένες παρουσιάσεις διαδικτυακών εφημερίδων δεν αργεί να ανακαλύψει την ανάγκη αυτή οπότε και δημοσιεύει το 1994, το πρώτο προσχέδιο των Cascading Style Sheets. Μεταξύ των επικριτών στο προσχέδιο αυτό, ήταν ο Dave Raggett (αρχιτέκτονας της HTML) και ο Bert Bos. Ο δεύτερος εκείνη την εποχή κατασκεύαζε ένα πρόγραμμα περιήγησης με style sheets , οπότε και ξεκινάει μια συνεργασία με τον Håkon [7]. Εκείνη την εποχή η CSS δεν ήταν η μοναδική πρόταση για γλώσσα stylesheet. Όμως κατάφερε να ξεχωρίσει καθώς είχε τη φιλοσοφία ότι ο σχεδιασμός του εγγράφου έπρεπε συνδυάζεται από τον δημιουργό αλλά και τον αναγνώστη. Πράγματι στα τέλη του 1995 το W3C (World Wide Web Consortium) δημιούργησε μια ομάδα που έδειξε μεγάλο ενδιαφέρον για την CSS και εργάστηκαν για να την μετατρέψουν σε σύσταση W3C. Ο στόχος αυτός επετεύχθη τελικά τον Δεκέμβριο του 1996 [6]. Η CSS2 συστάθηκε το 1998 αλλά υπέφερε από αρκετά σφάλματα. Μια μεταγενέστερη έκδοση της, ονόματι CSS2.1 έρχεται να διορθώσει τα σφάλματα της προηγούμενης .Η CSS2.1 προδιαγραφή τελικά δημοσιεύεται επίσημα ως σύσταση W3C το 2011. Το 1998 παράλληλα ξεκίνησαν και οι εργασίες για την CSS3 γνωρίζοντας βέβαια πως θα πάρει καιρό να κυκλοφορήσει επίσημα. Η CSS3 προδιαγραφή δεν κυκλοφορεί ενιαία αλλά σε ενότητες οι οποίες μάλιστα γράφονται με προδιαγραφές συμβατότητας με παλαιότερες εκδόσεις. Όποτε, όταν κάποια ενότητα θεωρείται stable παίρνει το καθεστώς υποψηφίας σύστασης CR(Candidate Recommendation) , το ίδιο μοτίβο ακολουθεί και η CSS4 δηλαδή οι ενότητες επιπέδου τέσσερα βελτιώνουν ενότητες του επιπέδου τρία ώστε να φτάσουν στην κατάσταση CR [8].

3.2.2 Η διάρθρωση της CSS

Η CSS (Cascading Style Sheets ή Επικαλυπτόμενα φύλλα στυλ) είναι μια γλώσσα υπεύθυνη για την εμφάνιση μιας σελίδας, χρησιμοποιείται δηλαδή για την διαχείριση της εμφάνισης εγγράφων html. Χωρίς την CSS μια ιστοσελίδα είναι μονότονη με βασικά χρώματα και διάταξη, κάτι που την κάνει βαρετή για τον αναγνώστη ίσως και δυσανάγνωστη. Στην Εικόνα 7 μπορούμε να δούμε ένα παράδειγμα για το πως περίπου θα ήταν η ιστοσελίδα της γνωστής εταιρίας amazon χωρίς CSS .



Εικόνα 7 : THE WEB WITHOUT CSS, ΙΣΤΟΣΕΛΙΔΑ https://dev.to

Η δομή της CSS αποτελείται από ένα σύνολο κανόνων, τα λεγόμενα φύλλα στυλ, αυτά διαμορφώνουν την εμφάνιση της σελίδας. Υπάρχουν δύο τρόποι για να δημιουργήσουμε φύλλα στυλ . Ο πρώτος είναι συγγραφή σε έναν text editor και ο δεύτερος είναι μέσω ειδικών εργαλείων τα οποία δεν απαιτούν την γνώση του συντακτικού της CSS. Εμείς σε αυτή την υποενότητα θα μιλήσουμε για τους κανόνες γραφής CSS σε έναν text editor, έτσι ώστε αρχικά κατανοήσουμε την δομή της, και έπειτα θα αναφερθούμε στα εργαλεία ανοιχτού κώδικα της Bootstrap που χρησιμοποιήθηκαν αρκετά στην εργασία μας.

Γενικά το συντακτικό της CSS είναι απλό. Ένας κανόνας αποτελείται από τον επιλογέα selector και την δήλωση (declaration). Ο επιλογέας ουσιαστικά επιλέγει ποιά στοιχεία html θα λάβουν τις αλλαγές στην εμφάνιση τους . Η δήλωση βρίσκεται πάντα μέσα σε αγκύλες και αποτελείται από δύο τμήματα, την ιδιότητα (property) και την τιμή (value), αυτά προσδιορίζουν τι είδους αλλαγή θα γίνει στο στοιχείο [9]. Ας δούμε ένα παράδειγμα για να το κατανοήσουμε καλύτερα : Έστω ότι θέλουμε να αλλάξουμε το χρώμα όλων των παραγράφων ενός εγγράφου σε κόκκινο, η εντολή που θα δώσουμε είναι **p { color :red }**. Όπως φαίνεται ο επιλογέας μας είναι το «p» που δείχνει πως η αλλαγή θα γίνει στα στοιχεία παραγράφων και μέσα στην δήλωση υπάρχει η ιδιότητα του χρώματος με τιμή «κόκκινο». Ο τύπος του selector καθορίζεται από το είδος των στοιχείων που στοχεύει κάθε φορά. Σύμφωνα με τον Powell [10, p. 471] έχουμε τους παρακάτω selectors :

1. **Επιλογέας Στοιχείου (Element Selector) :** Επιλέγει στοιχεία του ίδιου τύπου, δηλαδή στοιχεία που έχουν την ίδια HTML ετικέτα και παίρνει το όνομα του στοιχείου . Για

παράδειγμα έχουμε το στοιχείο με την ετικέτα μιας παραγράφου «<p>» , τότε ο επιλογέας μας θα είναι «p» . Αν επιθυμούμε να επιλέξουμε περισσότερα από ένα στοιχεία μπορούμε να διαμορφώσουμε τον επιλογέα μας σε «tag name1, tag name2, tag name3». Τέλος αν θέλουμε να εφαρμόσουμε έναν κανόνα σε όλα τα στοιχεία του εγγράφου μας χρησιμοποιούμε τον επιλογέα με σύμβολο «*».

2. **Επιλογέας Κλάσης (Class Selector) :** Επιλέγει στοιχεία που ανήκουν στην ίδια κλάση HTML, μπορεί ένα η περισσότερα στοιχεία να ανήκουν στην ίδια κλάση . Ο επιλογέας ορίζεται από το σύμβολο της τελείας («.») ακολουθούμενη από το όνομα της κλάσης. Για παράδειγμα αν έχουμε την κλάση με όνομα «myclass» ο επιλογέας διαμορφώνεται σε «.myclass»
3. **Επιλογέας ID (ID Selector) :** Επιλέγει στοιχεία με βάση το ID η αλλίως την ταυτότητα τους. Κάθε στοιχείο html έχει μοναδικό ID. Ο επιλογέας ορίζεται από το σύμβολο της δίεσης («#») και στη συνέχεια το όνομα του ID. Για παράδειγμα αν έχουμε το ID με όνομα «myid» τότε ο επιλογέας ορίζεται ως «#myid».
4. **Επιλογέας συμφραζομένων (Contextual Selector) :** Επιλέγει στοιχεία που βρίσκονται μέσα σε άλλα στοιχεία. Για παράδειγμα αν θέλουμε να επιβάλουμε μια αλλαγή στις ετικέτες «» που βρίσκονται μέσα σε μια παράγραφό μπορούμε να φτιάξουμε έναν επιλογέα : « p strong ».
5. **Επιλογέας χαρακτηριστικών (Attribute Selector) :** Αυτός ο επιλογέας στοχεύει στοιχεία με συγκεκριμένα χαρακτηριστικά. Όπως για παράδειγμα ο επιλογέας «a[href]» επιλέγει όλες τις ετικέτες «<a>» που έχουν το χαρακτηριστικό «href».

Με την CSS μπορούμε να ορίσουμε χρώματα ,γραμματοσειρές φόντο κτλπ. Αυτό καθορίζεται από τις ιδιότητες του εκάστοτε κανόνα. Παρακάτω θα αναφέρουμε μερικές από τις βασικές ιδιότητες css που αντλήσαμε από τον Powell [10, p. 542] .

Font : Ορίζει όλες τις ιδιότητες μια γραμματοσειράς. Στην κατηγορία αυτή ανήκουν ιδιότητες όπως font-size, font-weight, font-family κτλ.
Background: Ορίζει όλες τις ιδιότητες του φόντου. Υπάρχουν ιδιότητες όπως background-color, background image background-position κτλ.
Border : Ορίζει όλες τις ιδιότητες σχετικές με τα σύνορα ενός στοιχείου όπως border-width, border-style border-color κτλ.
Display : Καθορίζει τον τύπο εμφάνισης ενός στοιχείου
Margin : Αυτή η ιδιότητα ορίζει ένα περιθώριο σε όλες τις πλευρές ενός στοιχείου. Αν θέλουμε να ορίσουμε περιθώριο σε συγκεκριμένη πλευρά μπορούμε να ορίσουμε margin-top, margin-left κτλ.
Padding : Ορίζει ένα περιθώριο ανάμεσα στις πλευρές ενός στοιχείου και του περιεχομένου του.
Width: Ορίζει το πλάτος ενός στοιχείου.
Height : Ορίζει ύψος ενός στοιχείου.

Για να υλοποιηθεί οποιοσδήποτε κανόνας CSS σε ένα έγγραφο html πρέπει να δημιουργηθεί μια σύνδεση μεταξύ τους. Για την σύνδεση τους εφαρμόζουμε διάφορες τεχνικές οι οποίες παραθέτονται παρακάτω [9].

Εφαρμόζοντας τον κανόνα μέσα στο html αρχείο ανάμεσα στις ετικέτες «<style> </style>» του εγγράφου . Για παράδειγμα :

<head>
<style>
p { color : blue ; }
</style>
</head>

Ενσωματώνοντας τον κανόνα μέσα σε ένα στοιχείο html του εγγράφου χρησιμοποιώντας το χαρακτηριστικό «style». Για παράδειγμα :
< p style="color : blue;"> This is a blue paragraph </p>

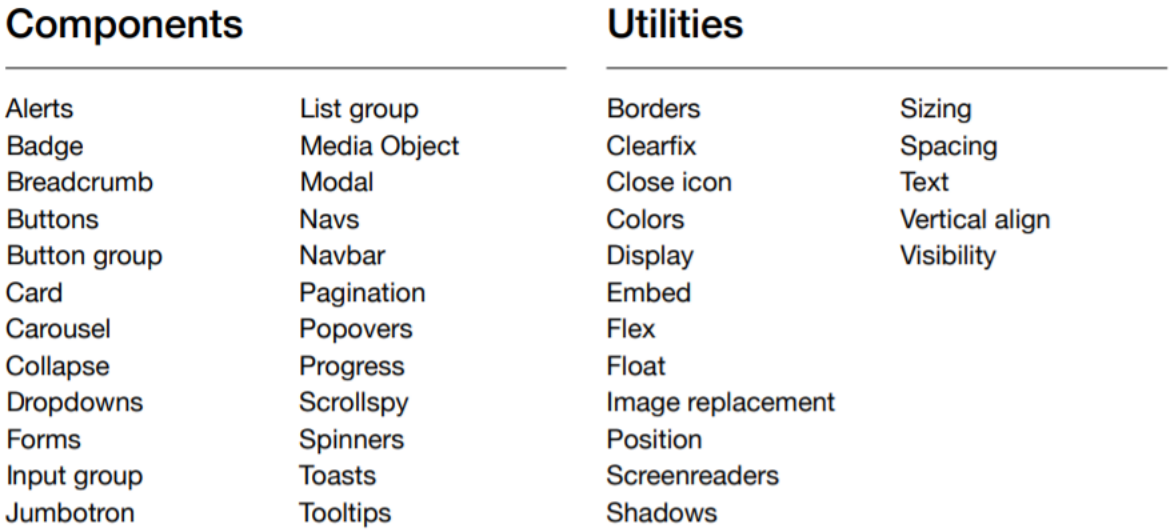
Γράφοντας τον κανόνα σε εξωτερικό αρχείο και συνδέοντας το με το html αρχείο χρησιμοποιώντας έναν σύνδεσμο. Για παράδειγμα :
<head>
<link rel = "stylesheet type="text/css" href="style.css">
</head>
Εισάγοντας ένα εξωτερικό αρχείο κανόνων χρησιμοποιώντας το @import notation. Για παράδειγμα :
@import "style.css";

3.2.3 Λίγα λόγια για την Bootstrap

Η Bootstrap είναι ένα δωρεάν, ελεύθερου λογισμικού framework το οποίο περιλαμβάνει διάφορα πρότυπα σχεδίασης βασισμένα σε HTML και CSS με σκοπό να διευκολύνει την ανάπτυξη ιστοσελίδων και διαδικτυακών εφαρμογών. Αυτή τη στιγμή είναι από τα πιο δημοφιλή CSS framework ανοιχτού κώδικα που υπάρχουν. Το έργο της Bootstrap ξεκίνησε στο twitter από τους Mark Otto και Jacob Thornton περίπου στα μέσα του 2010 και η αρχική της ονομασία ήταν Twitter Blueprint. Τον Αύγουστο του 2011 κυκλοφόρησε επίσημα ως εργαλείο ανοιχτού κώδικα. Η Bootstrap αποτελεί ένα δυνατό εργαλείο στα χέρια ενός προγραμματιστή και αυτό γιατί είναι εύκολη στη χρήση, μπορεί μέσω του responsive design να προσφέρει ευελιξία και δυνατότητα προσαρμογής μιας σελίδας σε οθόνες διαφόρων μεγεθών και τέλος γιατί είναι συμβατή με όλα τα σύγχρονα προγράμματα περιήγησης [11] .

Εικόνα 8 : BOOTSTRAP LOGO, ΙΣΤΟΣΕΛΙΔΑ https://commons.wikimedia.org/

Για να αποκτήσει κανείς πρόσβαση στην εργαλειοθήκη της Bootstrap αρκεί να κάνει λήψη των πακέτων της Bootstrap από την επίσημη σελίδα της (getbootstrap.com) ή αλλιώς να την ενσωματώσει στο έγγραφο του από ένα CDN (Content Delivery Network). Αφού αυτή η διαδικασία εκτελεστεί μπορεί πλέον να επωφεληθεί από τα διάφορα πρότυπα σχεδίασης που διαθέτει όπως προσχεδιασμένες φόρμες, κουμπιά, πίνακες, μπάρες πλοήγησης και άλλα [11]. Στην παρακάτω εικόνα μπορούμε να δούμε μια λίστα με τα διάφορα στοιχεία που αποτελούν την Bootstrap.



Εικόνα 9 : BOOTSRAP COMPONENTS AND UTILITIES, ΙΣΤΟΣΕΛΙΔΑ https://www.scmgalaxy.com/

Η Bootstrap αποτελείται από προκαθορισμένες κλάσεις με διάφορες επιλογές στυλ που μπορούμε να χρησιμοποιήσουμε εύκολα μέσα στο html έγγραφο μας. Σκεφτείτε ότι αν θέλουμε να κάνουμε μια αλλαγή στο στυλ ενός στοιχείου του εγγράφου μας, αντί να δημιουργούμε κανόνες CSS από την αρχή μπορούμε απλά να προσθέσουμε μια κλάση της Bootstrap η οποία θα έχει έτοιμο για εμάς τον επιθυμητό σχεδιασμό. Για παράδειγμα αν θέλουμε να αλλάξουμε το χρώμα μιας κεφαλίδας «<h1>» σε γκρι απλά προσθέτουμε την κλάση text-secondary στην κεφαλίδα και το χρώμα της αλλάζει.

3.2.4 Χρήση CSS και Bootstrap στην εφαρμογή μας

Στην υλοποίηση της εφαρμογής μας η χρήση «ατόφιας» CSS ήταν περιορισμένη λόγω της διευκόλυνσης μέσω της Bootstrap η οποία έκανε την περισσότερη δουλεία στο κομμάτι της διακόσμησης των σελίδων. Από την Bootstrap επωφεληθήκαμε με κάνοντας χρήση διάφορων κλάσεων της όπως buttons,modals, dropdowns, navigation bars, borders και displays και άλλα πολλά. Στην εικόνα 10 φαίνεται το dropdown menu που χρησιμοποιήθηκε για την ταξινόμηση των tickets ενώ στην εικόνα 11 ένα button το οποίο χρησιμοποιήθηκε για την αποσύνδεση του χρήστη και περιέχει επίσης ένα bootstrap icon.

```
44    <ul class="dropdown-menu">
45      <li><a class="dropdown-item"  href="order=DESC&collumn=time_created">Πιο πρόσφατα</a></li>
46      <li><a class="dropdown-item"  href="order=ASC&collumn=time_created">Πιο παλιά</a></li>
47      <li><a class="dropdown-item" href="order=ASC&collumn=ticket_solved">Κατάσταση (Ανοιχτά πρώτα)</a></li>
48
49    </ul>
```

Εικόνα 10

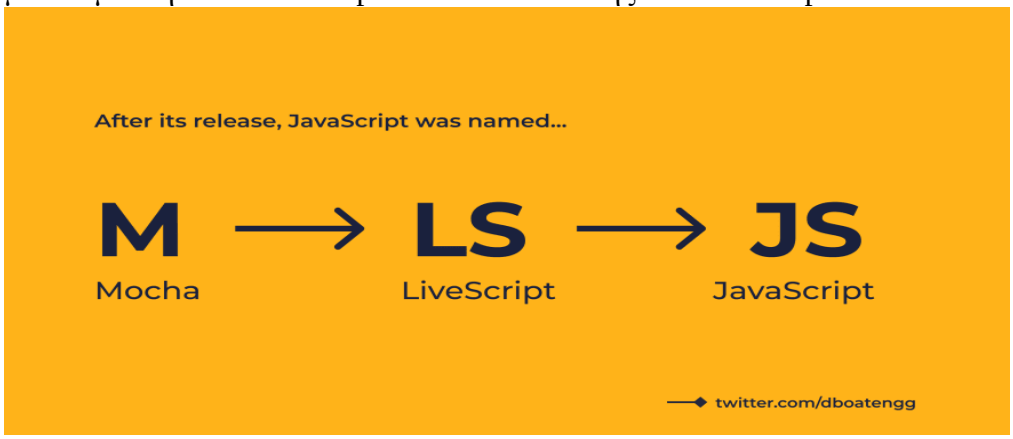
```
54    <button class='btn btn-outline-danger '  name='logout'><i class="bi bi-power"></i> <?php echo $_SESSION['name']; ?></button>
55
56
```

Εικόνα 11

3.3 JavaScript

3.3.1 Η ιστορία της JavaScript

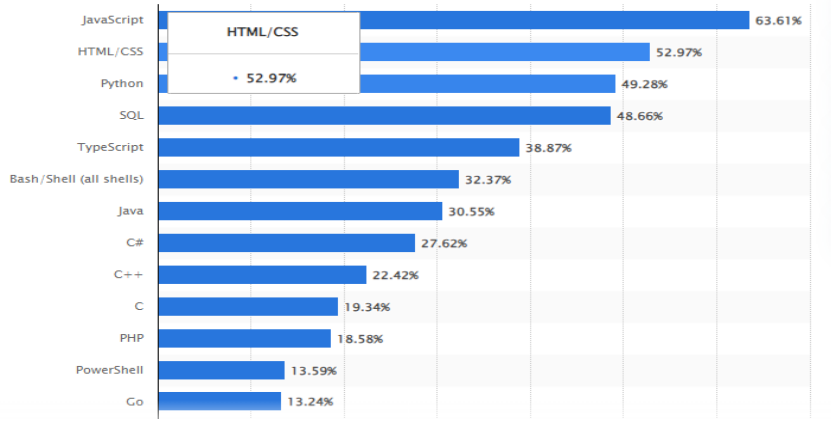
Η JavaScript είναι μία γλώσσα scripting η οποία δημιουργήθηκε για να παρέχει την δυνατότητα δυναμικής συμπεριφοράς σε μια ιστοσελίδα. Πριν την δημιουργία της δεν υπήρχε αυτή η δυνατότητα και οι ιστοσελίδες περιοριζόταν σε στατικό περιεχόμενο πράγμα που ώθησε την Netscape να δημιουργήσει μια γλώσσα που θα εξυπηρετεί τον σκοπό αυτό. Πράγματι το 1995 μέσα σε μόλις 10 μέρες ο Brendan Eich ,προγραμματιστής της Netscape, ανέπτυξε μια γλώσσα σεναρίου η οποία ξεκίνησε με το όνομα Mocha στη συνέχεια μετονομάστηκε σε Livescript και τελικά κατέληξε σε JavaScript.



Εικόνα 12 : Javascript names, ΙΣΤΟΣΕΛΙΔΑ https://dev.to/

Με την JavaScript δημιουργήθηκε μια σύγχυση καθώς πολλές εταιρίες δημιουργούσαν παραλλαγές της ώστε να δουλεύουν καλύτερα στους browser τους, το γεγονός αυτό δεν άργησε να φέρει ασυμβατότητες και δυσκολίες στους προγραμματιστές. Για αυτό τον λόγο οι

εταιρίες συμφώνησαν ότι πρέπει να χρησιμοποιούν κοινή γλώσσα ,την JavaScript. Οπότε η Netscape για να διασφαλίσει την υποστήριξη της δημιουργίας της, υπέβαλε την JavaScript στη Ευρωπαϊκή Ένωση Κατασκευαστών Υπολογιστών (ECMA) όπου και τυποποιήθηκε ως ECMAScript. Στις μέρες μας η JavaScript αποτελεί από τις πιο ευρέως χρησιμοποιημένες γλώσσες προγραμματισμού και χρησιμοποιείται στις περισσότερες ιστοσελίδες που υπάρχουν αυτή τη στιγμή στο διαδίκτυο, μεταξύ αυτών οι γνωστές σε όλους μας εφαρμογές YouTube και Facebook [12] .



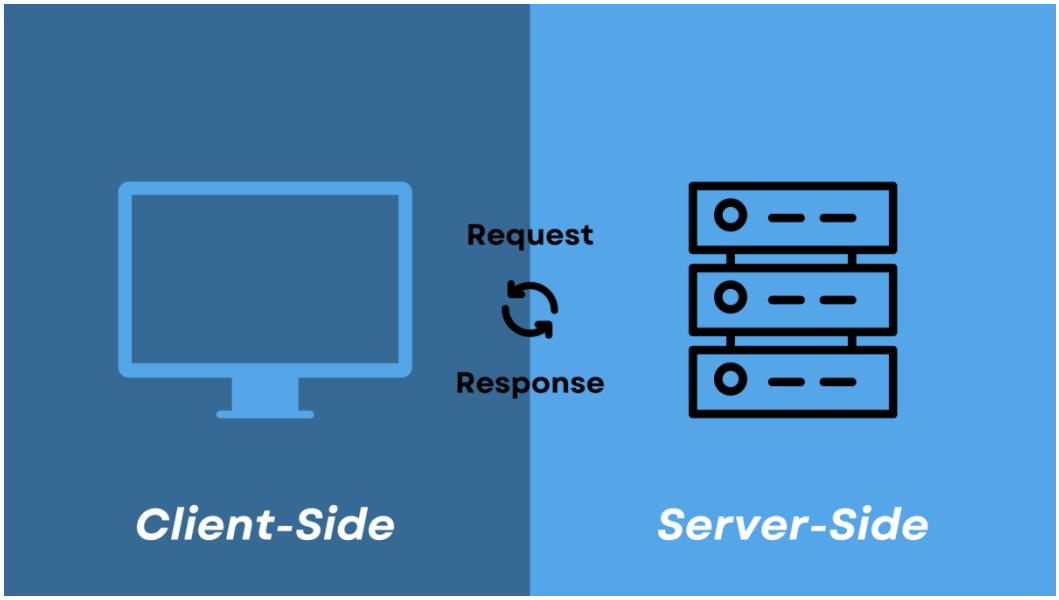
Εικόνα 13 : MOST USED PROGRAMMING LANGUAGES AMONG DEVELOPERS WORLDWIDE 2023, ΙΣΤΟΣΕΛΙΔΑ https://www.statista.com/

3.3.2 Η διάρθρωση της JavaScript

Η JavaScript είναι μια γλώσσα σεναρίων (scripting-language) υψηλού επιπέδου γνωστή για την ανάπτυξη ιστοσελίδων, από το όνομα της περιμένει κανείς πως η JavaScript σχετίζεται με τη Java, πράγμα που δεν ισχύει αν και το συντακτικό της μοιάζει με αυτό της Java. Ο πρωταρχικός λόγος που δημιουργήθηκε ήταν για να κάνει δυναμικό το περιεχόμενο μιας ιστοσελίδας. Οι εντολές της σε αντίθεση με άλλες γλώσσες προγραμματισμού πληκτρολογούνται δυναμικά, δεν χρειάζεται δηλαδή ο καθορισμός του τύπου μιας-μιας μεταβλητής εκ των προτέρων μάλιστα εκτελείται μία μόνο εντολή κάθε φορά. Η JavaScript περιέχει μια τυπική βιβλιοθήκη αντικειμένων και ένα βασικό σύνολο στοιχείων. Ανάλογα με το περιβάλλον της μπορεί να χρησιμοποιηθεί και από την πλευρά του πελάτη (client-side) για τον έλεγχο της συμπεριφοράς μιας ιστοσελίδας αλλά και την πλευρά του διακομιστή για επικοινωνία με την βάση δεδομένων [13].

Client-side: Η JavaScript αποκτά πρόσβαση στα στοιχεία ενός HTML εγγράφου χρησιμοποιώντας το DOM (Document Object Model) του. Αυτό της δίνει δυνατότητες όπως η αλλαγή στοιχείων σε φόρμες HTML και η ανταπόκριση σε συμβάντα χρήστη. Υπάρχουν βιβλιοθήκες της JavaScript όπως οι AngularJS, ReactJS και jQuery οι οποίες κάνουν πιο εύκολη την δουλεία αυτή.

Server-side: Η JavaScript μπορεί να εκτελείται στην πλευρά του διακομιστή και αυτό της επιτρέπει να χειρίζεται αρχεία μέσα σε αυτόν αλλά και να επικοινωνεί με την βάση δεδομένων ώστε να παρέχει πληροφορίες.



Εικόνα 14 : Client Side and Server Side Scripting, ΙΣΤΟΣΕΛΙΔΑ https://www.boardinfinity.com/

Για να συνδέσουμε την JavaScript με ένα html έγγραφο έχουμε διαθέσιμους δύο τρόπους. Ο πρώτος είναι να ενσωματώσουμε τον κώδικα της απευθείας στο έγγραφο ανάμεσα στις ετικέτες « <script> …. </script » (Internal JS). Ενώ ο δεύτερος τρόπος είναι να τρέξουμε τον κώδικα σε ένα αρχείο με κατάληξη « .js » και να τον συμπεριλάβουμε στο html έγγραφο μας μέσω ενός συνδέσμου (External JS).

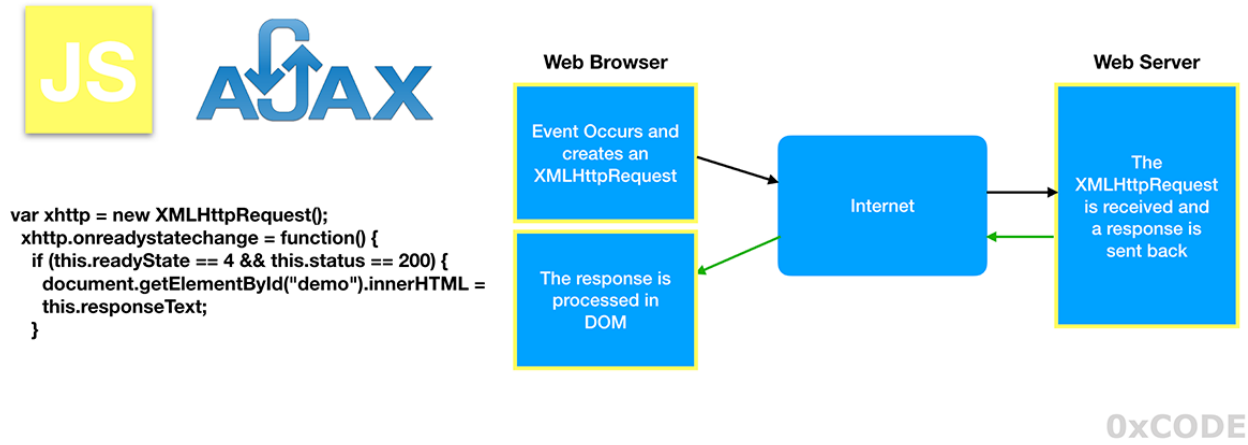
3.3.3 Λίγα λόγια για την JQuery

Η JQuery είναι μια βιβλιοθήκη της JavaScript η οποία χρησιμοποιείται για να απλοποιήσει τις αλληλεπιδράσεις ενός HTML/CSS εγγράφου. Για να χρησιμοποιήσουμε την βιβλιοθήκη αυτή χρειάζεται να κατεβάσουμε την κατάλληλη έκδοση απτο επίσημο site της (jQuery.com) η αλλιώς να χρησιμοποιήσουμε το κατάλληλο CDN. Η JQuery έχει εξυπηρετεί την φιλοσοφία “Write less, do more.” Για αυτό το λόγο μπορεί να επιλέγει απευθείας τα στοιχεία με CSS selectors και να κάνει αλλαγές πάνω τους μέσω των μεθόδων της [14]. Μπορούμε να δείξουμε ένα παράδειγμα που αντιπροσωπεύει πλήρως αυτή την φράση, το παράδειγμα μας θα είναι ένα “hello World” με JavaScript και JQuery αντίστοιχα :
JavaScript : document.getElementById("myid").innerHTML = "Hello, World!";
JQuery: $("#myid").html("Hello, World!");

Η βιβλιοθήκη της JQuery δίνει πολλούς λόγους σε έναν προγραμματιστή για να την χρησιμοποιήσει στο πρόγραμμα του [14] :
1. Μεγάλη γκάμα πρόσθετων που αναβαθμίζονται συνεχώς.
2. Μεγάλη αναπτυξιακή κοινότητα.
3. Εύκολη στην χρήση.
4. Απλοποιεί τα AJAX (Asynchronous JavaScript And XML) πρότυπα για ασύγχρονη επικοινωνία με τον διακομιστή κάτι που χρησιμοποιήθηκε και βοήθησε σημαντικά στην υλοποίηση της συγκεκριμένης εργασίας, επομένως αξίζει να αναλυθεί περεταίρω στην παρακάτω υποενότητα.

3.3.4 AJAX (Asynchronous JavaScript and XML)

Η AJAX χρησιμοποιείται στην ανάπτυξη διαδικτυακών εφαρμογών, μέσω των τεχνολογιών XHTML, CSS, JavaScript, DOM, XML, XSLT και XMLHttpRequest επιτυγχάνει την δυναμική αλληλεπίδραση ιστοσελίδας-χρήστη. Πριν την δημιουργία της AJAX, οι ιστότοποι αποτελούνταν από διάφορες σελίδες συνδεδεμένες μεταξύ τους μέσω links οπότε κάθε φόρα που ο χρήστης αλληλεπιδρούσε με τον ιστότοπο έπρεπε να περάσει κάποιος χρόνος ώστε να φορτωθεί κάθε σελίδα ολόκληρη με αποτέλεσμα την αργοπορημένη ανταπόκριση στο αίτημα του επομένως και την ανάλογη αναμονή. Στις 15 Φεβρουαρίου ο Jesse James Garrett παρουσίασε μια καινοτομία η οποία έδινε λύση στο παραπάνω πρόβλημα, την AJAX. Αυτή η καινοτομία άνοιξε το δρόμο της ασύγχρονης επικοινωνίας μεταξύ ιστοσελίδας και server [15].
Η λειτουργία της AJAX επιτρέπει σε κάθε αίτημα του χρήστη, να μεταφέρονται μόνο οι απαιτούμενες πληροφορίες από και προς τον διακομιστή χωρίς να χρειάζεται εκ νέου φόρτωση ολόκληρης της σελίδας. Αυτό σημαίνει ότι η επικοινωνία προγράμματος περιήγησης και διακομιστή εκτελείται ασύγχρονα στο παρασκήνιο χωρίς να διακόπτει την εκτέλεση της εφαρμογής. Ως αποτέλεσμα, προκύπτουν διαδραστικές εφαρμογές ιστού που προσφέρουν γρήγορη απόκριση και βελτιωμένη απόδοση. Το αποτύπωμα στον χρήστη είναι η καλύτερη εμπειρία περιήγησης και ο μικρός χρόνο αναμονής [15].



Εικόνα 15 :AJAX - Asynchronous Javascript And XML, ΙΣΤΟΣΕΛΙΔΑ https://medium.com/

3.3.5 Χρήση JavaScript στην εφαρμογή μας

Στην εφαρμογή μας χρησιμοποιήσαμε την βιβλιοθήκη JQuery της JavaScript κυρίως για να δημιουργήσουμε αιτήματα AJAX ώστε αλλάζουμε δυναμικά το περιεχόμενο της σελίδας χωρίς να ανανεώνεται ολόκληρη. Επίσης χρησιμοποιήθηκε και για εμφάνιση μηνυμάτων σφάλματος η επιτυχίας σε υποβολές φόρμας. Στην παρακάτω εικόνα φαίνεται η συνάρτηση «loadUserData» η οποία εκτελεί ένα αίτημα jQuery AJAX ώστε να φορτώσει τα δεδομένα του χρήστη σε μορφή JSON και έπειτα να τα εκτυπώσει μέσω της «printUserData».

```
function loadUserData(pageNr) {

    $.ajax({
    type: 'GET',
    url: 'user_actions.php',
    data: { 'page-nr': pageNr },
    dataType: 'json',

    success: function (response) {
        $('#userTable tbody').find('tr:gt(0)').remove();


        $.each(response.data, function (index, item) {

            printUserData(item);


            });
        },
    });
}

```

Εικόνα 16

3.4 PHP (Hypertext Preprocessor)

Εικόνα 17 : PHP-logo, ΙΣΤΟΣΕΛΙΔΑ https://en.m.wikipedia.org/

3.4.1 Η ιστορία της PHP

Το 1994 ο Rasmus Lerdorf δημιούργησε την πρώτη υλοποίηση της PHP, η ονομασία της προερχόταν από τα αρχικά της φράσης «Personal Home Page Tools». Αρχικά ήταν μια υλοποίηση λίγο διαφορετική από αυτή που ξέρουμε σήμερα καθώς αποτελούνταν από ένα σύνολο δυαδικών αρχείων γραμμένα στη γλώσσα προγραμματισμού C που χρησιμοποιήθηκε από τον Rasmus για την παρακολούθηση των επισκέψεων στο διαδικτυακό βιογραφικό του. Αργότερα, εμπλουτίστηκε με νέες λειτουργίες όπως η αλληλεπίδραση με την βάση δεδομένων και στη συνέχεια με μια ολική επανεγγραφή του κώδικα της, απέκτησε δομή παρόμοια με αυτή της C. Η νέα αυτή έκδοση ονομάστηκε PHP/ FI 2.0 (Forms Interpreter) και πλέον περιλάμβανε λειτουργίες χρήστη, cookies αλλά και υποστήριξη για βάσεις δεδομένων DBM, mSQL και Postgres95 [16].

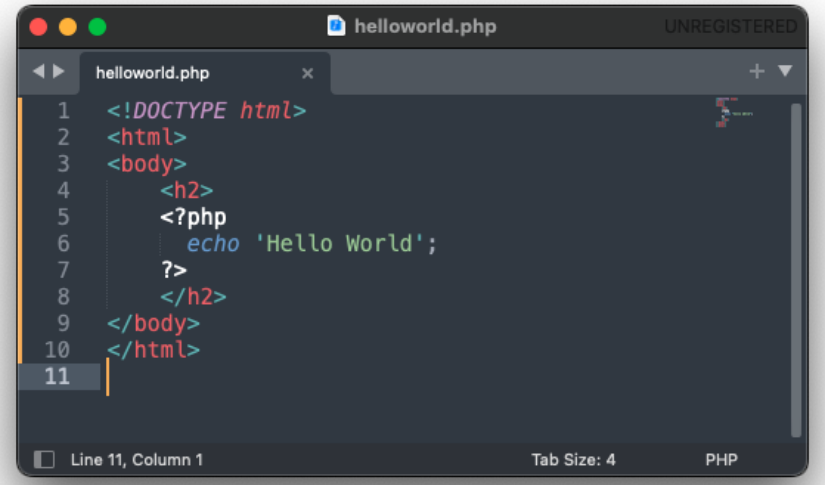
Το 1997 ο Andi Gutmans και ο Zeev Suraski χτίζοντας πάνω στην προηγούμενη έκδοση αποφάσισαν ότι πρέπει να φτιαχτεί μια νέα ανεξάρτητη γλώσσα προγραμματισμού όπως και έκαναν. Αφαίρεσαν τον δεσμευτικό τίτλο PHP/FI και την μετονομάσαν απλώς σε PHP έκδοση 3.0 επίσης αναδιαμόρφωσαν τον όρο σε PHP «Hypertext Preprocessor». Η PHP 3.0 ήταν και η πρώτη έκδοση που θυμίζει την σημερινή υλοποίηση, χαρακτηριστικα που την έκαναν να ξεχωρίζει ήταν η επεκτασιμότητα της και η υποστήριξη αντικειμενοστρέφειας, επιπρόσθετα σημαντική ήταν και η παροχή ώριμης διεπαφής στους χρήστες για βάσεις δεδομένων, πρωτόκολλα και API. Το 1998 ο Andi Gutmans και ο Zeev Suraski υλοποίησαν

ένα καινούριο σχεδιασμό με αλλαγές στην απόδοση σύνθετων εφαρμογών και βελτίωση της αρθρωτής βάσης κώδικα. Η PHP 4.0 τελικά ,επιτυχαίνοντας τους παραπάνω στόχους, κυκλοφορεί επισήμως το 2000 με πυρήνα το «Zend Engine». Τέλος κάπου προς το καλοκαίρι του 2004 έπειτα απο μεγάλη ανάπτυξη κυκλοφορεί η PHP 5.0 με αρκετά νέα χαρακτηριστικά αλλά διατηρώντας τον ίδιο πυρήνα [16].

3.4.2 Η διάρθρωση της PHP

Η PHP είναι μία γλώσσα προγραμματισμού σεναρίων η οποία εκτελείται στην πλευρά του διακομιστή και είναι κατάλληλη για την ανάπτυξη ιστού. Ο κώδικας PHP μπορεί να ενσωματωθεί σε ένα HTML έγγραφο μέσω ειδικών ετικετών. Με την PHP μπορούμε να κάνουμε διάφορες ενέργειες όπως να δημιουργούμε δυναμικού περιεχομένου σελίδες, να συλλέγουμε δεδομένα από φόρμες, να διαχειριζόμαστε βάσεις δεδομένων και να αποστέλλουμε cookies. Ο κώδικας της PHP τρέχει στο server ενώ στον πελάτη επιστρέφονται μόνο τα αποτελέσματα της εκτέλεσης του. Για να τρέξει ένα πρόγραμμα PHP χρειαζόμαστε ένα πρόγραμμα περιήγησης ιστού έναν PHP parser και τον server. Βέβαια υπάρχει η δυνατότητα να τρέξει και στην γραμμή εντολών, σε αυτή τη περίπτωση χρειάζεται μόνο ο parser ώστε να εκτελέσει απευθείας τον κώδικα.

Η PHP, όπως κάθε γλώσσα προγραμματισμού, έχει το δικό της συντακτικό. Είναι απλή γλώσσα με ρίζες σε C και Perl. Όταν ο διακομιστής συναντήσει αρχείο με κατάληξη «.php» αναγνωρίζει αυτομάτως ότι πρόκειται για αρχείο που περιέχει κώδικα γραμμένο σε PHP. Ο κώδικας ξεκινάει με την ετικέτα «<?php» η οποία σηματοδοτεί την αρχή στον server ώστε να ερμηνεύσει και να εκτελέσει τον κώδικα μέχρι και την ετικέτα τερματισμού «?>», οτιδήποτε είναι μετά από αυτή την ετικέτα απλά αγνοείται από τον Parser και αποστέλλεται ως HTML στον πελάτη. Κάθε εντολή της PHP πρέπει να τελειώνει με το σύμβολο «;» ώστε να γίνεται ξεκάθαρο το πότε αρχίζει η επόμενη. Όλες οι μεταβλητές της PHP θα πρέπει να ξεκινάνε με το σύμβολο «$», αυτό βοηθάει στον Parser να τις αντιλαμβάνεται αμέσως. Μάλιστα ο τύπος κάθε μεταβλητής, αναγνωρίζεται αυτόματα από την τιμή που θα της εκχωρήσουμε οπότε δεν είναι αναγκαία η δήλωση πριν από την χρήση της. Τέλος πρέπει να γνωρίζουμε ότι οι μεταβλητές της PHP είναι case-sensitive [17]. Στην παρακάτω εικόνα θα δούμε ενδεικτικά έναν απλό κώδικα «Hello world» σε PHP.



Εικόνα 18 : PHP Hello World, ΙΣΤΟΣΕΛΙΔΑ https://www.tutorialkart.com/

3.4.3 Μερικά χαρακτηριστικά της PHP

Η PHP δίνει ελευθερία στην επιλογή λογισμικού καθώς μπορεί να χρησιμοποιηθεί στα περισσότερα λειτουργικά συστήματα όπως Linux, MacOs και Windows, την ίδια ελευθερία παρέχει και στην επιλογή server με μια ευρεία υποστήριξη στους περισσότερους διακομιστές ιστού που κυκλοφορούν. Ακόμη παρέχει ευελιξία ως προς τον τρόπο προγραμματισμού, μπορεί να χρησιμοποιηθεί για αντικειμενοστραφή προγραμματισμό (OOP), για διαδικαστικό η και τα δύο. Τέλος ένα από τα πιο σημαντικά χαρακτηριστικά αυτής της γλώσσας είναι η υποστήριξη της για ένα μεγάλο εύρος βάσεων δεδομένων [18]. Παρακάτω φαίνονται κάποιες μόνο από τις βάσεις δεδομένων που υποστηρίζει.

Εικόνα 19 : Best Five Databases To Use With PHP, ΙΣΤΟΣΕΛΙΔΑ https://blog.singsys.com/

3.4.4 Χρήση της PHP στην εφαρμογή μας

Το μεγαλύτερο κομμάτι στην εφαρμογή μας περιλαμβάνει εντολές php. Εντολές υπεύθυνες για την αλληλεπίδραση με την βάση, για την επεξεργασία δεδομένων και άλλων λειτουργιών. Για να μπορεί ο διακομιστής να εκτελέσει τις εντολές της PHP πρέπει το αρχείο να έχει κατάληξη «.php» για αυτό και στην εφαρμογή μας θα δείτε αρχεία όπως database.php, init.php, actions.php, index.php και άλλα πολλά. Το καθένα από αυτά εκτελεί σημαντικές λειτουργίες καθοριστικές για την λειτουργία της εφαρμογής μας. Για παράδειγμα στο αρχείο database.php ορίζεται η κλάση database στην οποία εκτελείται η σύνδεση με την βάση δεδομένων MySQL (Εικόνα 20). Στο αρχείο init.php γίνεται η αρχικοποίηση των κλάσεων με την δημιουργία αντικειμένων των αντίστοιχων κλάσεων (Εικόνα 21) έτσι ώστε να μπορούμε να καλέσουμε τις μεθόδους κάθε κλάσης. Τέλος στην Εικόνα 22 μπορούμε να δούμε την μέθοδο «logUserIn» της κλάσης user η οποία συνδέει τον χρήστη στην εφαρμογή.

classes > database.php

```php
<?php
class Database {

    private $host;
    private $dbname;
    private $username;
    private $password;

    public function __construct(){
        $this->host = "localhost";
        $this->dbname = "ticketing system";
        $this->username = "root";
        $this->password = "";

        try {

            $this->conn = new PDO ("mysql:host=$this->host;dbname=$this->dbname",$this->username,$this->password);

        }

        catch(Exception $e){
                    echo "Connection failed: " . $e->getMessage();
        }
    }


}



?>
```

Εικόνα 20

init.php

```php
<?php

session_start();
 require 'classes/database.php';
 require 'classes/user.php';
 require 'classes/department.php';
 require 'classes/Ticket.php';
 require 'classes/reply.php';


$database = new Database();
$user = new User();
$department = new Department();
$ticket = new Ticket();
$reply = new Reply();


 ?>
```

Εικόνα 21

```php
public function  logUserIn () {

    if (!empty($_POST["password"]) && !empty($_POST["username"]))   {

        $name= $this->validate($_POST['username']);
        $password= $this->validate($_POST['password']);
        $sql = "SELECT * from $this->userTable WHERE name=:name AND password=:password";
        $stmt= $this->conn->prepare($sql);
        $stmt->bindParam(':name',$name);
        $stmt->bindParam(':password',$password);
        $stmt->execute();

        if  ($row = $stmt->fetch()) {
            if ( $row['name']===$name && $row['password']=== $password ) {

                $_SESSION['name'] = $name;
                $_SESSION['id'] = $row['id'];
                $_SESSION['status'] = $row['status'];
                $_SESSION['user_type']= $row['user_type'];
                header('Location: ../tickets.php');
                $sql = "UPDATE $this->userTable  SET status=true WHERE id ='".$_SESSION["id"]."'";
                $this->conn->query($sql);
            }
        }
    }
}
```

Εικόνα 22

3.5 Βάσεις δεδομένων/ Διαχείριση βάσεων

3.5.1 Λίγα λόγια για τις βάσεις δεδομένων

Ως βάση δεδομένων ορίζουμε μία συλλογή από σχετιζόμενα δεδομένα η σχετιζόμενες πληροφορίες που συνήθως αποθηκεύονται σε ηλεκτρονική μορφή σε ένα υπολογιστικό σύστημα. Για τον χειρισμό μίας βάσης δεδομένων χρειάζεται ένα σύστημα διαχείρισης βάσεων δεδομένων (RDBMS) ενώ για την σύνταξη και τα ερωτήματα δεδομένων, η γλώσσα SQL (Structured Query Language). Οι βάσεις δεδομένων ξεκίνησαν για πρώτη φορά την δεκαετία του 1960, αρχικά οι βάσεις που αναπτύχθηκαν ήταν οι ιεραρχικές βάσεις και οι βάσεις δικτύου στην συνέχεια αναπτύχθηκαν και οι σχεσιακές βάσεις οι οποίες είναι οι πιο κοινές στις μέρες μας και χρησιμοποιούν πίνακες για την ομαδοποίηση των δεδομένων τους [19].

3.5.2 SQL (Structured Query Language)

Η SQL είναι μια γλώσσα προγραμματισμού ή αλλιώς μια γλώσσα ερωτημάτων κατάλληλη για την αλληλεπίδραση με σχεσιακές βάσεις δεδομένων. Συγκεκριμένα μπορεί να εκτελεί λειτουργίες όπως η αποθήκευση, ο χειρισμός και η ανάκτηση δεδομένων σε μια βάση δεδομένων. Αναπτύχθηκε την δεκαετία του 1970 από τον Edgar Frank "Ted" Codd για την IBM και το 1986 έγινε πρότυπο στο εθνικό ινστιτούτο ANSI (American National Standards Institute), ένα χρόνο μετά έλαβε και πιστοποίηση ISO (International Organization for Standardization). Η SQL είναι η τυπική γλώσσα για όλα τα συστήματα διαχείρισης βάσης δεδομένων (RDBMS) όπως MySQL, MS Access, Oracle, Postgres και SQL Server. Ο τρόπος λειτουργίας τους έχει ως εξής : όταν δίνουμε μια εντολή SQL για κάποιο από τα παραπάνω συστήματα, οι μηχανές εκτέλεσης SQL αποφασίζουν τον καλύτερο τρόπο να ερμηνεύσουν την εργασία που αναθέσαμε [20].

3.5.3 MySQL

Εικόνα 23 : MySQL logo, ΙΣΤΟΣΕΛΙΔΑ https://wikitech.wikimedia.org/

Η MySQL είναι ένα σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων το οποίο χρησιμοποιεί την γλώσσα SQL για την επικοινωνία με την βάση. Αναπτύχθηκε από την Oracle, είναι από τα πιο δημοφιλή συστήματα διαχείρισης βάσεων και είναι ανοικτού κώδικα. Σκεφτείτε ότι μια βάση δεδομένων μπορεί να αποτελείται από τεράστιο όγκο πληροφοριών, για να αποκτήσουμε πρόσβαση και να κάνουμε αλλαγές σε αυτές τις πληροφορίες είναι απαραίτητο ένα RDBMS σαν αυτό της MySQL. Η λειτουργία της ξεκινά με την δημιουργία μιας βάσης δεδομένων όπου για την εκτέλεση των διαφόρων λειτουργιών οι χρήστες υποβάλλουν αιτήματα, ο server ανταποκρίνεται στα αιτήματα και τα εμφανίζει στην πλευρά του χρήστη. Οι MySQL βάσεις είναι σχεσιακές, αυτό σημαίνει ότι οργανώνουν τα δεδομένα σε δομές πίνακα με γραμμές και στήλες αντί να τα στοιβάζουν όλα μαζί. Υπάρχουν διάφορα συστήματα διαχείρισης βάσεων δεδομένων παρακάτω θα δούμε μερικά απο τα πλεονεκτήματα της MySQL που μας ώθησαν στην επιλογή της [21].

- Είναι δωρεάν, απλώς την κατεβάζουμε από τον επίσημο ιστότοπο της και είναι έτοιμη προς χρήση.
- Μπορεί να χειριστεί τεράστιο όγκο δεδομένων.
- Ανταποκρίνεται με γρήγορες ταχύτητες και μεγάλη απόδοση.
- Είναι συμβατή σε διάφορα λειτουργικά συστήματα
- Παρέχει μεγάλη ασφάλεια δεδομένων καθώς και κρυπτογράφηση κωδικών.
- Χρησιμοποιεί την αρχιτεκτονική πελάτη –διακομιστή.

3.5.4 phpMyAdmin

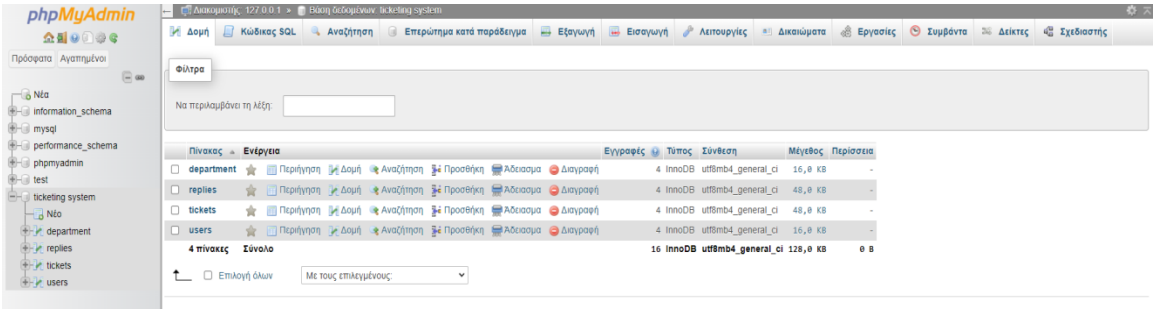
Εικόνα 24 : PhpMyAdmin logo, ΙΣΤΟΣΕΛΙΔΑ https://en.m.wikipedia.org/

Το PhpMyAdmin αποτελεί ένα εργαλείο λογισμικού ανοιχτού κώδικα γραμμένο σε PHP και είναι κατάλληλο για την διαχείριση της MySQL στο διαδίκτυο. Ενώ δίνει την δυνατότητα εκτέλεσης οποιασδήποτε εντολής σε SQL, αποτελεί ένα UI (User Interface) εργαλείο, δηλαδή μπορεί η διαχείριση την βάσης δεδομένων και διάφορες άλλες λειτουργίες να εκτελούνται εύκολα μέσα από ένα γραφικό περιβάλλον για τον χρήστη. Παρακάτω θα αναφέρουμε κάποιες από τις κύριες δυνατότητες που μας παρέχει [22].

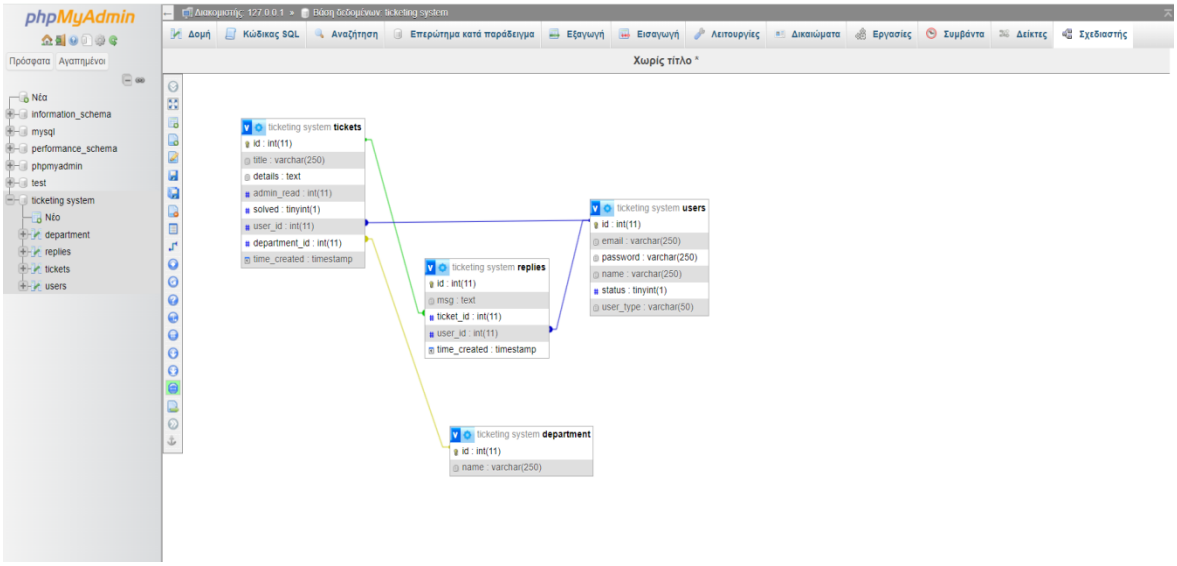
- Υποστήριξη για πολλές από τις δυνατότητες της MySQL όπως : διαγραφή και δημιουργία βάσεων δεδομένων, επεξεργασία πινάκων πεδίων και ευρετηρίων, συντήρηση της βάσης και εκτέλεση SQL ερωτημάτων .
- Εισαγωγή CSV και SQL δεδομένων.
- Εξαγωγή δεδομένων σε μορφές CSV, SQL, XML, PDF, Word και LATEX.
- Διαχείριση πολλαπλών διακομιστών.
- Δημιουργία σύνθετων ερωτημάτων.
- Δημιουργία γραφικών από την βάση σε διάφορες μορφές.

3.5.5 Η διαχείριση της βάσης στην εφαρμογή μας

Για την διαχείριση της βάσης μας χρησιμοποιήθηκε το εργαλείο PhpMyAdmin. Μέσα από το γραφικό περιβάλλον που μας παρείχε δημιουργήθηκαν οι κατάλληλοι πίνακες και οι μεταξύ τους συσχετίσεις. Στην εικόνα 25 μπορούμε να δούμε τους πίνακες που αποτελούν την βάση για το σύστημα μας ενώ στην εικόνα 26 μπορούμε να δούμε σχεσιακό μοντέλο της βάσης μας.



Εικόνα 25



Εικόνα 26

3.6 APACHE Server /XAMPP

Εικόνα 27 : XAMPP, ΙΣΤΟΣΕΛΙΔΑ https://www.acewebacademy.com/

3.6.1 Λίγα λόγια για το εργαλείο XAMPP

Το XAMPP αποτελεί ένα πακέτο λογισμικού ανοιχτού κώδικα που περιλαμβάνει το εργαλείο PHPMyAdmin και τις λειτουργικές μονάδες Apache HTTP Server, MYSQL, MariaDB, PHP και Perl. Αναπτύχθηκε από την εταιρία APACHE και είναι διαθέσιμο σε έντεκα διαφορετικές γλώσσες με υποστήριξη σε διάφορες πλατφόρμες. Το XAMPP βοηθά τους προγραμματιστές να τρέχουν το πρόγραμμα τους τοπικά (localhost) ώστε να κάνουν δοκιμές σε αυτό πριν το ανεβάσουν στον κύριο διακομιστή. Είναι μια πλατφόρμα η οποία παρέχει ένα πλήρες περιβάλλον ανάπτυξης για την επαλήθευση και τις δοκιμές πάνω στην ανάπτυξη διαδικτυακών εφαρμογών. Παρακάτω αναφέρουμε κάποια από τα συστατικά του [23].

- **Cross-platform:** Είναι διαθέσιμο για διάφορα λειτουργικά συστήματα και πλατφόρμες όπως οι διανομές Windows, MacOS και Linux.
- **Apache:** Είναι ένας διακομιστής HTTP ο οποίος είναι δωρεάν και χρησιμοποιείται απο τους προγραμματιστές για την παράδοση περιεχομένων ιστού.
- **MySQL,MariaDb:** Είναι συστήματα διαχείρισης σχεσιακών βάσεων δεδομένων (DBMS) τα οποία προσφέρουν υπηρεσίες επεξεργασίας δεδομένων. Οι νεότερες εκδόσεις του xampp χρησιμοποιούν πλέον ως προεπιλογή το MariaDB αντί της MySQL.
- **PHP:** Είναι μία γλώσσα προγραμματισμού σεναρίων για την ανάπτυξη ιστού.
- **PERL:** Είναι μια γλώσσα προγραμματισμού η οποία χρησιμοποιείται για την ανάπτυξη ιστού και για να δίνει λύσεις σε προβλήματα διαχείρισης συστήματος.
- **phpMyAdmin:** Εργαλείο κατάλληλο για την διαχείριση της MySQL και MariaDB στο διαδίκτυο
- **OpenSSL:** Είναι υλοποίηση του Secure Socket Layer Protocol.
- **Xampp Control Panel:** Είναι το κέντρο ελέγχου του Xampp
- **Filezila:** Είναι ένας διακομιστής πρωτοκόλλου μεταφοράς αρχείων.
- **Mercury:** Είναι ένα σύστημα μεταφοράς αλληλογραφίας.
- **Tomcat:** Είναι ένα εργαλείο που παρέχει Java λειτουργίες.

3.6.2 Ο διακομιστής Apache

Εικόνα 28 : Apache Server logo, ΙΣΤΟΣΕΛΙΔΑ https://en.m.wikipedia.org/

Ο Apache HTTP Server είναι ένας δωρεάν και ανοιχτού κώδικα διακομιστής ιστού ο οποίος αναπτύχθηκε και συντηρείται από την Apache Software Foundation, η πρώτη του κυκλοφορία έγινε το 1995. Αυτή τη στιγμή, αποτελεί τον πιο δημοφιλή διακομιστή στο

διαδίκτυο και χρησιμοποιείται από μεγάλες εταιρίες όπως Cisco, IBM, Facebook και EBay. Τέλος είναι cross-platform με υποστήριξη τόσο σε Unix όσο και σε windows συστήματα και χρησιμοποιεί το πρωτόκολλο HTTP για την επικοινωνία πελάτη-διακομιστή.

Ένας διακομιστής μπορεί να παραδίδει ιστοσελίδες μέσω του διαδικτύου, βασική του λειτουργία να δέχεται αιτήματα χρήστη και να στέλνει απαντήσεις σε αυτά. Ο Apache δεν είναι ένας φυσικός διακομιστής, αλλά είναι ένα λογισμικό που τρέχει στον διακομιστή, παρόλα αυτά έχει οριστεί ως διακομιστής. Στόχος του Apache είναι να συνδέει το πρόγραμμα περιήγησης με τον διακομιστή, όταν για παράδειγμα ένας επισκέπτης θέλει να φορτώσει μια σελίδα του ιστότοπου μας το πρόγραμμα περιήγησης του θα στείλει ένα αίτημα στον διακομιστή μας. Ο Apache τότε επιστρέφει μια απάντηση στο αίτημα με τα ζητούμενα αρχεία. Κάποια από τα πλεονεκτήματα χρήσης του Apache είναι τα εξής [24]:

- Αποτελεί σταθερό και αξιόπιστο λογισμικό.
- Είναι δωρεάν και ανοιχτού κώδικα.
- Δέχεται συνεχείς ενημερώσεις ασφαλείας.
- Είναι εύκολο και φιλικό για τον χρήστη
- Βρίσκει υποστήριξη σε πλατφόρμες Windows και Unix.

Κεφάλαιο 4: Παρουσίαση εφαρμογής

Σε αυτό το κεφάλαιο θα παρουσιάσουμε την λειτουργία και τα σενάρια χρήσης της εφαρμογής από την πλευρά του διαχειριστή αλλά και απτήν πλευρά του απλού χρήστη με τρόπο που θα δώσει στον αναγνώστη να καταλάβει πλήρως την φιλοσοφία της. Πριν ξεκινήσουμε θα αναφέρουμε τις λειτουργίες χρήστη που εκτελούνται.

1. **Διαχειριστής :**

- Προβολή της λίστας χρηστών, tickets και κτηρίων.
- Δημιουργία, διαγραφή, επεξεργασία λογαριασμού χρηστών.
- Εισαγωγή και διαγράφη κτηρίων.
- Δημιουργία ,επεξεργασία, διαγραφή οποιουδήποτε ticket.
- Ορισμός ticket ως «κλειστό» η «ανοικτό» ανάλογα.
- Εκτέλεση όλων των λειτουργιών που αναφέρονται στον κοινό χρήστη.

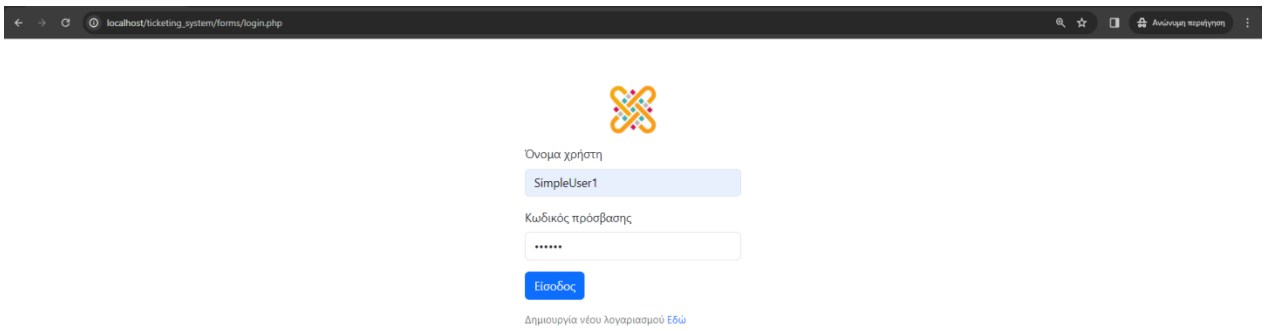
2. **Κοινός χρήστης**

- Προβολή της λίστας των ticket που έχει δημιουργήσει ο ίδιος ο χρήστης.
- Επεξεργασία των στοιχείων του λογαριασμού του όπως αλλαγή κωδικού η username.
- Δημιουργία ticket, επεξεργασία και διαγραφή στα ticket που έχουν δημιουργηθεί από τον ίδιο.
- Συνομιλία με τον διαχειριστή για πληροφόρηση σχετικά με την πορεία της βλάβης στην συνομιλία του εκάστοτε ticket.

4.1 Δημιουργία αιτήματος βλάβης και συνομιλία (Διαχειριστής & Κοινός Χρήστης)

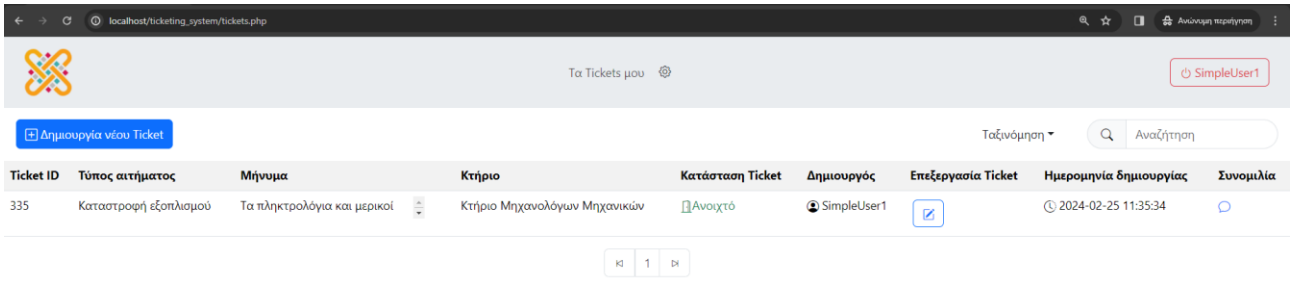
Σε αυτή την υποενότητα παρουσιάζεται κάθε βήμα δημιουργίας ενός αιτήματος βλάβης και απεικονίζονται φωτογραφίες για την πλήρη κατανόηση. Η διαδικασία και για τους δύο τύπους χρήστη είναι κοινή οπότε η παρακάτω περιγραφή, ενώ γίνεται από την μεριά ενός απλού χρήστη, αντιπροσωπεύει την διαδικασία και για τους δύο χρήστες.

Βήμα 1: Εισαγωγή διαπιστευτηρίων για είσοδο στην εφαρμογή (Εικόνα 29).



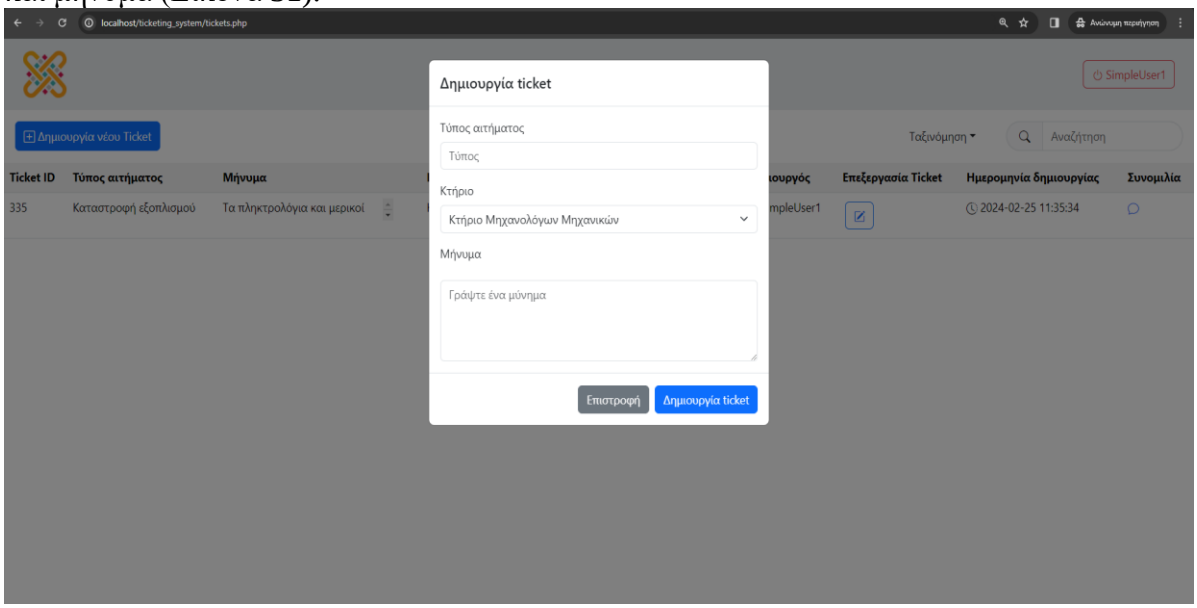
Εικόνα 29

Βήμα 2: Επιλογή πλήκτρου δημιουργίας νέου Ticket στο αριστερό και πάνω μέρος του κέντρου ελέγχου (Εικόνα 30).



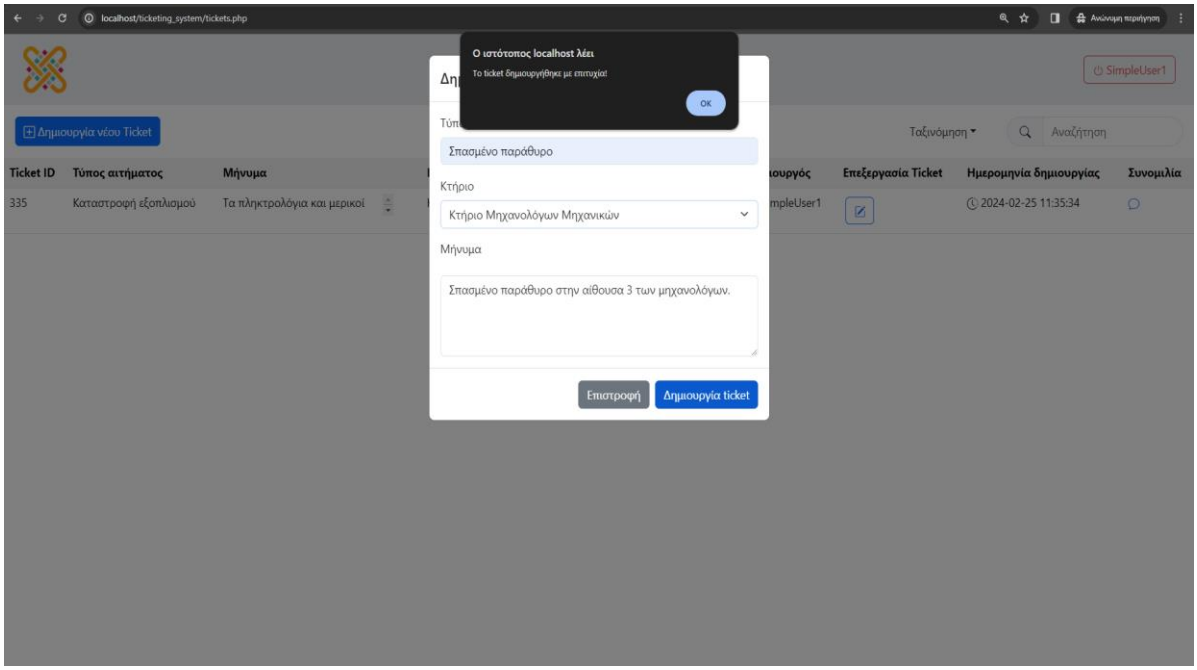
Εικόνα 30

Βήμα 3: Συμπλήρωση φόρμας αιτήματος βλάβης όπου ζητούνται : Τύπος αιτήματος, Κτήριο και μήνυμα (Εικόνα 31).



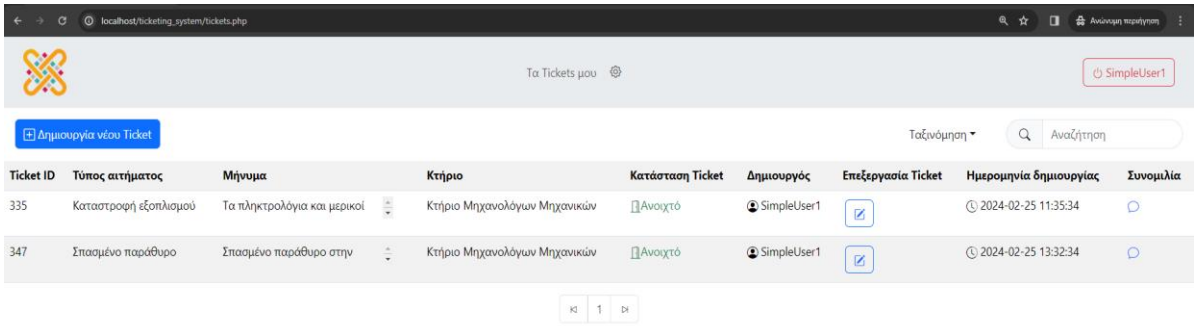
Εικόνα 31

Βήμα 4: Κλικ στο κουμπί «Δημιουργία Ticket» και ένδειξη μηνύματος επιτυχούς δημιουργίας (Εικόνα 32) .



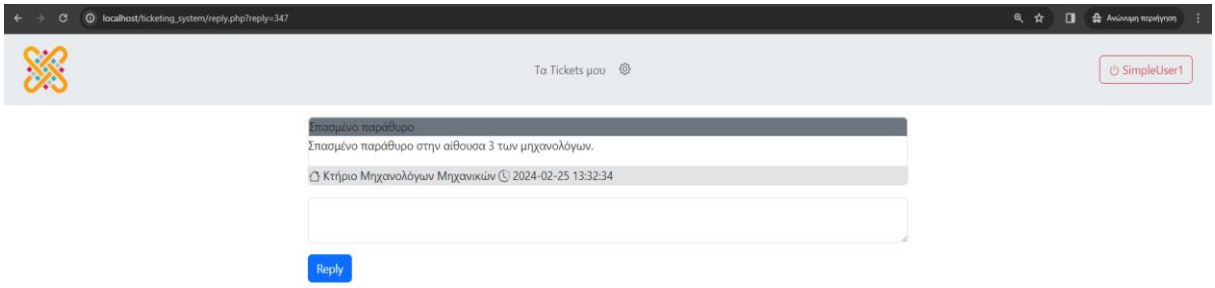
Εικόνα 32

Βήμα 5: Επιλέγοντας «OK» στο μήνυμα επιτυχίας η λίστα με τα tickets ανανεώνεται και το νέο ticket εμφανίζεται πλέον στην λίστα οπού παρέχονται οι επιλογές συνομιλίας και επεξεργασίας Ticket και χαρακτηριστικά όπως : ID, τύπος αιτήματος, μήνυμα, κτήριο, κατάσταση, δημιουργός και ημερομηνία δημιουργίας (Εικόνα 33).



Εικόνα 33

Βήμα 6: Από τη στιγμή που έχει δημιουργηθεί το ticket κάνοντας κλικ στο εικονίδιο της συνομιλίας μπορούμε να ανοίξουμε την συνομιλία για επικοινωνία μεταξύ διαχειριστή-χρήστη και ενημέρωση για την εξέλιξη του αιτήματος (Εικόνα 34).

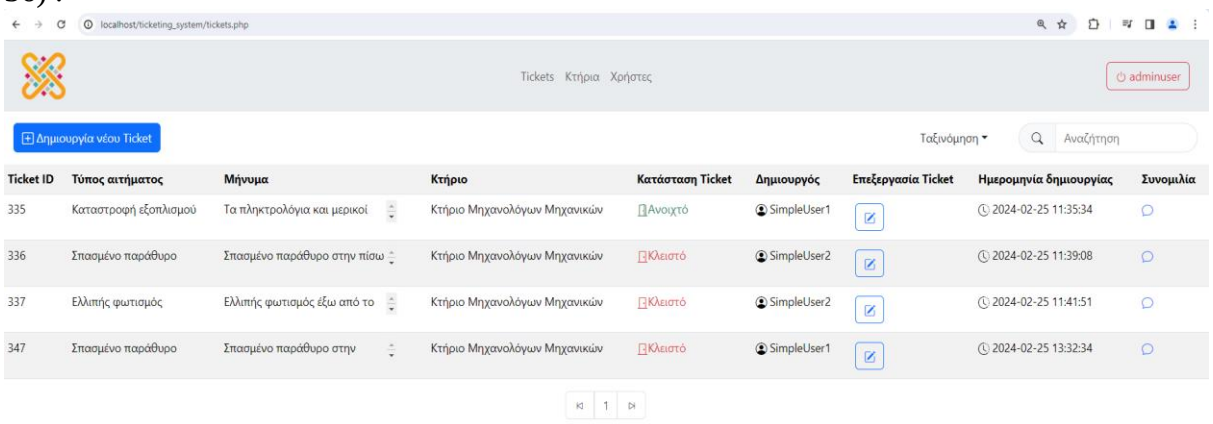


Εικόνα 34

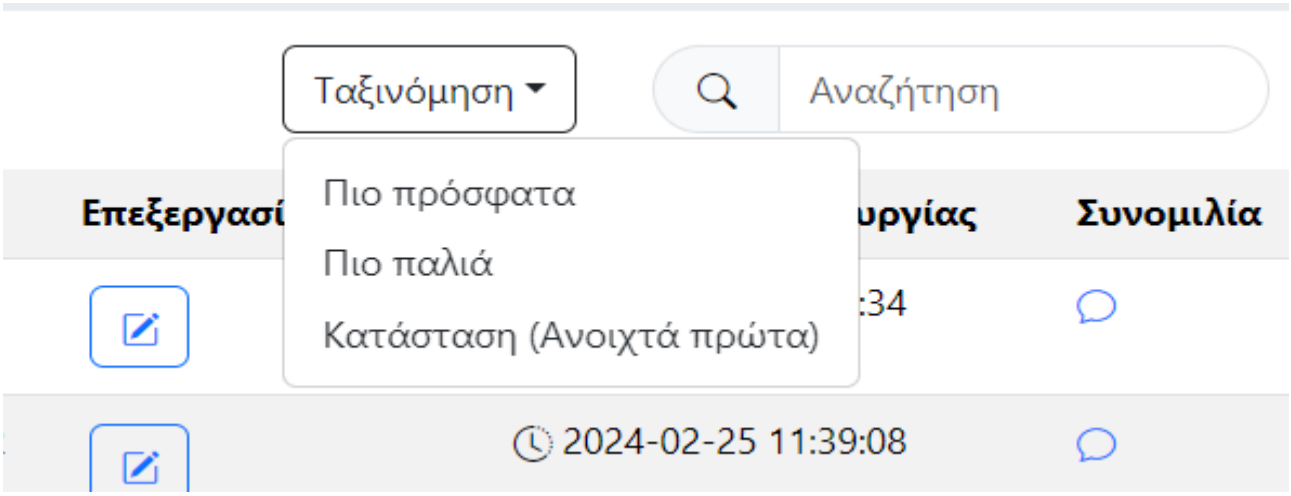
4.2 Διαχείριση ticket (Διαχειριστής).

Βήμα 1: Εισαγωγή διαπιστευτηρίων για είσοδο στην εφαρμογή (όπως Εικόνα 29).

Βήμα 2: Έλεγχος της λίστας αιτημάτων και εντοπισμός των εκκρεμών (ανοιχτών) tickets (Εικόνα 35). Μπορεί επίσης να γίνει ταξινόμηση με βάση τα ανοικτά πρώτα . Η μέσω της αναζήτησης να ψάξει συγκεκριμένο αίτημα σύμφωνα με το id,το κτήριο ή το μήνυμα (Εικόνα 36) .

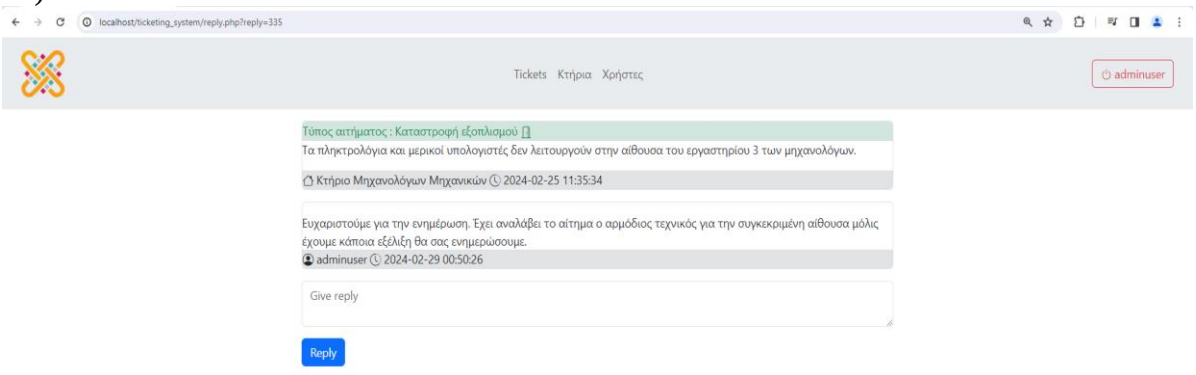


Εικόνα 35



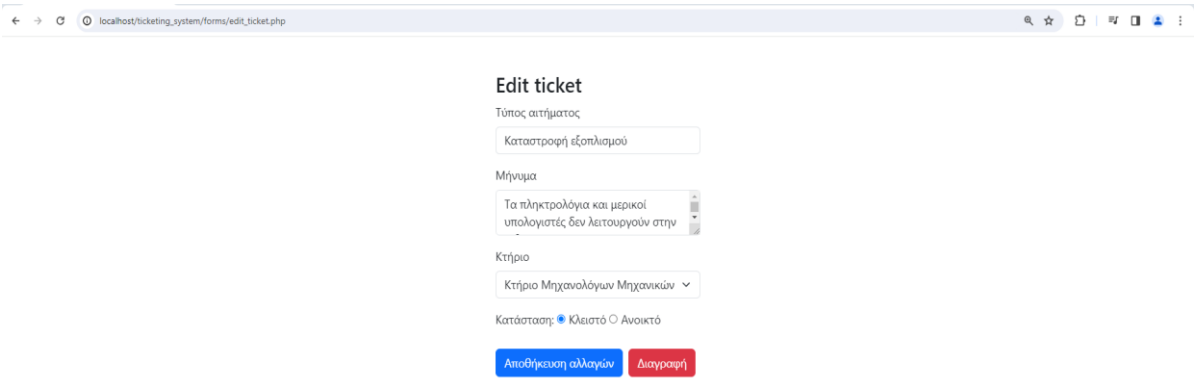
Εικόνα 36

Βήμα 3: Επικοινωνία με τον χρήστη και ενημέρωση για την εξέλιξη του αιτήματος (Εικόνα 37).

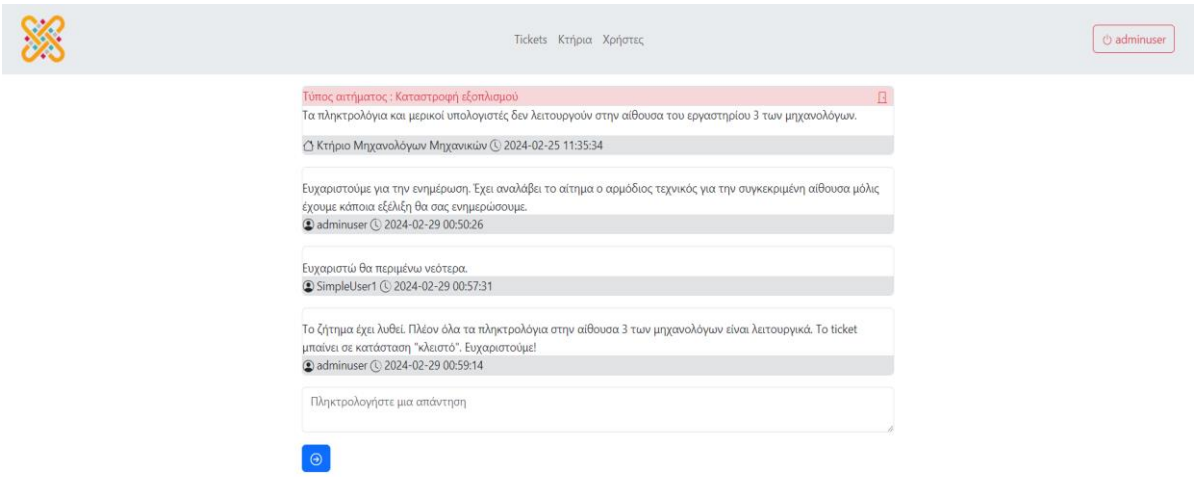


Εικόνα 37

Βήμα 4: Αφού το αίτημα ανατεθεί στον αρμόδιο και επιλυθεί, ακολουθεί η σήμανση του ticket ως κλειστό και η αποστολή μηνύματος επιβεβαίωσης στον χρήστη (Εικόνες 38 και 39).



Εικόνα 38



Εικόνα 39

Κεφαλαίο 5: Επίλογος

5.1 Συμπεράσματα –Περιορισμοί

Στην παρούσα διπλωματική εργασία σχεδιάστηκε μια διαδικτυακή εφαρμογή με σκοπό να εξυπηρετήσει τις ανάγκες ενός πανεπιστημίου στην καταγραφή και διαχείριση των βλαβών των κτηρίων του. Στο πλαίσιο ανάπτυξης της εφαρμογής έγινε εκμάθηση διάφορων εργαλείων και γλωσσών προγραμματισμού για τα οποία αφιερώθηκε αρκετός χρόνος ενασχόλησης και προσωπικής μελέτης. Η πορεία ανάπτυξης ήταν μια νέα εμπειρία με νέες δυσκολίες και εμπόδια κάτι που μας προσέφερε μια καλή γνωριμία με τον κόσμο του προγραμματισμού διαδικτύου.

Τα συμπεράσματα στα οποία καταλήξαμε ήταν:

- Η υλοποίηση του κώδικα ήταν ένα δύσκολο και άγνωστο κομμάτι το οποίο μας έμαθε την σημασία της μάθησης και προσαρμογής σε νέες τεχνολογίες.
- Η εφαρμογή μπορεί να μην έχει λειτουργήσει σε πραγματικές συνθήκες αλλά φαίνεται πως έχει την δυνατότητα (ίσως με κάποιες προσθήκες η προσαρμογές) να εξυπηρετήσει πλήρως τις ανάγκες ενός πανεπιστημίου.
- Η εφαρμογή που έχει αναπτυχθεί είναι φιλική προς τον χρήστη και απλή στην λειτουργία της.
- Ο βασικός περιορισμός στην υλοποίηση αυτού του έργου έχει να κάνει με το γεγονός ότι δεν δοκιμάστηκε σε πραγματικές συνθήκες. Αυτό μας περιόρισε στον εντοπισμό πιθανών σφαλμάτων και πραγματικής αξιολόγησης της ευχρηστίας της. Ιδανικά θα έπρεπε να γίνουν δοκιμές σε πραγματικές συνθήκες λειτουργίας στα πλαίσια του πανεπιστημίου και να έχουμε ανατροφοδότηση από τους χρήστες ώστε να προσαρμόσουμε κατάλληλα την λειτουργία της εφαρμογής.

5.2 Μελλοντικές επεκτάσεις

Η ανάπτυξη του συστήματος μας δεν σταματάει εδώ. Υπάρχουν βελτιώσεις και μελλοντικές επεκτάσεις οι οποίες μπορούν να ανοίξουν νέες λειτουργίες και δυνατότητες. Πολύ χρήσιμη φαίνεται πως θα ήταν η δημιουργία εφαρμογής για κινητές συσκευές ios και android μάλιστα θα μπορούσε στην υπάρχων πλατφόρμα να γίνει μια καλύτερη διεπαφή χρήστη. Επιπλέον θα ήταν καλό να ενσωματωθεί στην εφαρμογή μας και μία άλλη κατηγορία χρηστών αυτή των τεχνικών επίλυσης των βλαβών οι οποίοι θα χωρίζονται σε κατηγορίες ανάλογα με την αρμοδιότητα τους και θα έχουν απευθείας πρόσβαση στο αντίστοιχο αίτημα βλάβης. Αυτό θα διευκόλυνε το έργο του διαχειριστή ο οποίος πρέπει τώρα να διαχωρίζει τις βλάβες χειροκίνητα και να τις αναθέτει αυτός στους αρμόδιους. Τέλος κάτι πιο φιλόδοξο, θα μπορούσε με την βοήθεια της τεχνητής νοημοσύνης και της μηχανικής μάθησης να ενσωματωθεί ένα σύστημα πρόβλεψης των επερχόμενων βλαβών και εξαγωγής στατιστικών στοιχείων.

Παραρτήματα

Παρακάτω επισυνάπτονται μερικά από τα σημαντικά κομμάτια κώδικα που δημιουργήθηκαν για τις ανάγκες της εργασίας μας.

Κλάσεις -Μέθοδοι.

Database.php

```
   <?php
class Database {

   private $host;
   private $dbname;
   private $username;
   private $password;

   public function __construct(){
      $this->host = "localhost";
      $this->dbname = "ticketing system";
      $this->username = "root";
      $this->password = "";

      try {

         $this->conn = new PDO ("mysql:host=$this->host;dbname=$this->dbname",$this->username,$this-
>password);

      }

      catch(Exception $e){
                  echo "Connection failed: " . $e->getMessage();
      }

   }

}




?>
```

User.php

```
      <?php


class User Extends Database {
 private $userTable;

   public function __construct(){
      parent::__construct();
      $this->userTable='users';


   }

   public function createAccount($name,$password,$email,$password2) {
      $name = $this->validate($name);
```

```
        $password = $this->validate($password);
        $email = $this->validate($email);
        $password2 = $this->validate($password2);

        $sql = "SELECT * FROM $this->userTable WHERE email = :email";
        $stmt=$this->conn->prepare($sql);
        $stmt->bindParam(':email',$email);
        $stmt->execute();

            if (!$row = $stmt->fetch()) {
                                    if    (!empty($_POST["email"])    &&    !empty($_POST["password"])    &&
!empty($_POST["username"]))   {
                    if($password === $password2){

                                $sql = "INSERT INTO $this->userTable (name, password, email) VALUES (:name,
:password, :email)";
                        $stmt=$this->conn->prepare($sql);
                        $stmt->bindParam(':name',$name);
                        $stmt->bindParam(':password',$password);
                        $stmt->bindParam(':email',$email);
                        $stmt->execute();

                        echo "<script>";
                        echo "alert('Ο χρήστης δημιουργήθηκε με επιτυχία');";
                        echo "</script>";

                    }
                }
            }

            else {
                echo "<script>";
                echo "alert('This email already exists');";
                echo "</script>";

            }
        }


    public function editAccount($id) {

                            if      (!empty($_POST["email"])            &&      !empty($_POST["username"])      &&
!empty($_POST["password"]))   {

                $sql = "UPDATE $this->userTable SET name = :name, email = :email, password= :password,
user_type=:user_type WHERE id = :id";
            $stmt=$this->conn->prepare($sql);
            $stmt->bindParam(':name',$_POST['username']);
            $stmt->bindParam(':email',$_POST['email']);
            $stmt->bindParam(':password',$_POST['password']);
            $stmt->bindParam(':user_type',$_POST['usertype']);
            $stmt->bindParam(':id',$id);
            $stmt->execute();

                if ($_SESSION['user_type'] === 'admin') header('Location: ../users.php');
```

```
            else  header('Location: ../tickets.php');

        }

        else echo "Write all fields";

    }
    public function validate($data){

        $data = trim($data);
        $data = stripslashes($data);
        $data = htmlspecialchars($data);
        return $data;

    }

    public function  logUserIn () {

        if (!empty($_POST["password"]) && !empty($_POST["username"]))   {

            $name= $this->validate($_POST['username']);
            $password= $this->validate($_POST['password']);
            $sql = "SELECT * from $this->userTable WHERE name=:name AND password=:password";
            $stmt= $this->conn->prepare($sql);
            $stmt->bindParam(':name',$name);
            $stmt->bindParam(':password',$password);
            $stmt->execute();

            if  ($row = $stmt->fetch()) {
                if ( $row['name']===$name && $row['password']=== $password ) {

                    $_SESSION['name'] = $name;
                    $_SESSION['id'] = $row['id'];
                    $_SESSION['status'] = $row['status'];
                    $_SESSION['user_type']= $row['user_type'];
                    header('Location: ../tickets.php');
                    $sql = "UPDATE $this->userTable  SET status=true WHERE id ='".$_SESSION["id"]."'";
                    $this->conn->query($sql);
                }
            }
        }
    }

    public function isLoggedIn () {

        return isset($_SESSION['id']);
    }

    public function logUserOut () {

        $sql = "UPDATE $this->userTable  SET status=false WHERE id ={$_SESSION["id"]}";
        $this->conn->query($sql);
        header('Location: forms/login.php');
        session_destroy();
```

```
    }

    public function returnAllUsersData () {

        $start=0;
        $rows_per_page=4;

        if (isset($_GET['page-nr'])) {
            $page=$_GET['page-nr'] - 1;
            $start= $page * $rows_per_page;
        }

        $sql="SELECT * from $this->userTable LIMIT $start,$rows_per_page" ;
        $data = $this->conn->query($sql);
        $list = $data->fetchAll(PDO::FETCH_ASSOC);
        return json_encode(['data' => $list], JSON_PRETTY_PRINT);

    }


    public function returnUserData($id) {

        $sql = "SELECT * from $this->userTable WHERE id=$id";
        $data = $this->conn->query($sql);
        return $row = $data->fetch();

    }

    public function deleteUser($idfordelete) {

        $sql= "DELETE FROM $this->userTable WHERE id = $idfordelete";
        $this->conn->query($sql);
        header('Location: ../users.php');

    }




    public function returnPages() {

        $rows_per_page=4;
        $sql="SELECT * from $this->userTable" ;
        $data = $this->conn->query($sql);
        $num_of_rows=$data->rowCount();
        return $pages= ceil($num_of_rows / $rows_per_page);

    }
}



?>
```

Ticket.php

```
<?php

class Ticket extends Database {
   private $ticketTable;
   private $userId;


   public function __construct () {

      parent::__construct();
      $this->ticketTable='tickets';
      if (isset($_SESSION['id']))  $this->userId=$_SESSION['id'];

   }

   public function returnTicketTableData() {

      $start = 0;
      $rows_per_page = 4;


      if (!empty($_GET['page-nr'])) {
         $page = (int)$_GET['page-nr'] - 1;
         $start = $page * $rows_per_page;
         $_SESSION['start']=$start;
      }

      if (isset($_GET["order"]) && isset($_GET["collumn"])) {
         $order = $_GET['order'];
         $collumn = $_GET['collumn'];
         $_SESSION['order']=$order;
         $_SESSION['collumn']=$collumn;
      }


      if ($_SESSION['user_type'] === 'admin') {


         if (isset($_SESSION['order']) && isset($_SESSION['collumn'])) {
               $sql = "SELECT ur.name AS user_name, t.id AS ticket_id, t.title AS ticket_title, t.details AS
ticket_details, t.time_created AS time_created, t.admin_read AS admin_read, t.solved AS ticket_solved,
dp.name AS department_name
                  FROM $this->ticketTable t
                  JOIN department dp ON t.department_id = dp.id
                  JOIN users ur ON t.user_id = ur.id
                  ORDER BY ".$_SESSION['collumn']." ".$_SESSION['order']."
                  LIMIT ".$_SESSION['start'].", $rows_per_page";
         }

         else {
```

```
            $sql = "SELECT ur.name AS user_name, t.id AS ticket_id, t.title AS ticket_title, t.details AS ticket_details, t.time_created AS time_created, t.admin_read AS admin_read, t.solved AS ticket_solved, dp.name AS department_name
                    FROM $this->ticketTable t
                    JOIN department dp ON t.department_id = dp.id
                    JOIN users ur ON t.user_id = ur.id
                    LIMIT ".$_SESSION['start']."", $rows_per_page";
        }

        $stmt = $this->conn->prepare($sql);
    }

    else  {

        if (isset($_SESSION['order']) && isset($_SESSION['collumn'])) {
            $sql = "SELECT ur.name AS user_name, t.id AS ticket_id ,t.title AS ticket_title, t.details AS ticket_details,t.time_created AS time_created, t.admin_read AS admin_read, t.solved AS ticket_solved, dp.name AS department_name
                    FROM $this->ticketTable t
                    JOIN department dp ON t.department_id = dp.id
                    JOIN users ur ON t.user_id = ur.id
                    WHERE  t.user_id=:userid
                    ORDER BY ".$_SESSION['collumn']." ".$_SESSION['order']."
                    LIMIT $start ,$rows_per_page";
          $stmt = $this->conn->prepare($sql);


        }

        else  {
            $sql = "SELECT ur.name AS user_name, t.id AS ticket_id ,t.title AS ticket_title, t.details AS ticket_details,t.time_created AS time_created, t.admin_read AS admin_read, t.solved AS ticket_solved, dp.name AS department_name
                    FROM $this->ticketTable t
                    JOIN department dp ON t.department_id = dp.id
                    JOIN users ur ON t.user_id = ur.id
                    WHERE  t.user_id=:userid
                    LIMIT $start ,$rows_per_page";
          $stmt = $this->conn->prepare($sql);
        }

        $stmt->bindParam(':userid',$this->userId);

    }

    $stmt->execute();
    $list = $stmt->fetchAll(PDO::FETCH_ASSOC);
    return json_encode(['data' => $list], JSON_PRETTY_PRINT);



}
```

```
    public function editTicketData($id){

    $sql = "UPDATE $this->ticketTable t
            JOIN department dp ON t.department_id = dp.id
            SET t.details = :details, t.title = :title, t.solved = :solved, t.department_id = :department
            WHERE t.id = :id";

    $stmt = $this->conn->prepare($sql);
    $stmt->bindParam(':details',  $_POST['details']);
    $stmt->bindParam(':title', $_POST['title']);
    $stmt->bindParam(':solved', $_POST['solved']);
    $stmt->bindParam(':department', $_POST['department']);
    $stmt->bindParam(':id', $id);
    $stmt->execute();
    header('Location: ../tickets.php');
}

    public function returnTicketData($id) {

        $sql = "SELECT  t.id AS ticket_id ,t.title AS ticket_title, t.details AS ticket_details, t.admin_read AS
admin_read,t.time_created AS time_created, t.solved AS ticket_solved, dp.name AS department_name
                FROM $this->ticketTable t
                JOIN department dp ON t.department_id = dp.id
                WHERE t.id = :id";

        $stmt = $this->conn->prepare($sql);
        $stmt->bindParam(':id', $id);
        $stmt->execute();
        return $row = $stmt->fetch();
    }


    public function returnPages() {

        $rows_per_page=4;
            $sql = "SELECT ur.name AS user_name, t.id AS ticket_id ,t.title AS ticket_title, t.details AS
ticket_details, t.admin_read AS admin_read, t.solved AS ticket_solved, dp.name AS department_name
                FROM $this->ticketTable t
                JOIN department dp ON t.department_id = dp.id
                JOIN users ur ON t.user_id = ur.id";

        if (  $_SESSION['user_type']==='admin') {
            $stmt = $this->conn->prepare($sql);

        }

        else   {

            $userSql = $sql . " WHERE t.user_id=:userid";
            $stmt = $this->conn->prepare($userSql);
            $stmt->bindParam(':userid',$this->userId);
        }

        $stmt->execute();
        $num_of_rows=$stmt->rowCount();
```

```
        return $pages= ceil($num_of_rows / $rows_per_page);

    }

    public function createTicket($subject,$details,$department) {

            $sql = "INSERT INTO $this->ticketTable (title,details,user_id,department_id) VALUES ( :subject,
:details, :userId, :department)";
            $stmt=$this->conn->prepare($sql);
            $stmt->bindParam(':subject',$subject);
            $stmt->bindParam(':details',$details);
            $stmt->bindParam(':userId',$this->userId);
            $stmt->bindParam(':department',$department);
            $stmt->execute();


    }




    function chooseDepartment() {

        $sql="SELECT * from department";
        $data=$this->conn->query($sql);

        while($row=$data->fetch())  {
            echo "<option    name='department' value='" . $row['id'] . "' >" . $row['name'] . "</option>";

        }
    }

    function deleteTicket($id) {

    $sql="DELETE FROM $this->ticketTable WHERE id= :id ";
    $stmt=$this->conn->prepare($sql);
    $stmt->bindParam(':id',$id);
    $stmt->execute();
    header('Location: ../tickets.php');

    }

    function searchTicket($search) {

        if ($_SESSION['user_type'] === 'admin') {
              $sql = "SELECT ur.name AS user_name, t.id AS ticket_id ,t.title AS ticket_title, t.details AS
ticket_details, t.admin_read AS admin_read, t.solved AS ticket_solved, dp.name AS department_name
                  FROM $this->ticketTable t
                  JOIN department dp ON t.department_id = dp.id
                  JOIN users ur ON t.user_id = ur.id
                   WHERE  t.id like :search or t.title like :search or ur.name like :search or t.details like :search or
dp.name like :search ";

            $stmt=$this->conn->prepare($sql);
```

```
        }

        else {
            $sql = "SELECT ur.name AS user_name, t.id AS ticket_id ,t.title AS ticket_title, t.details AS ticket_details,t.time_created AS time_created, t.admin_read AS admin_read, t.solved AS ticket_solved, dp.name AS department_name
                FROM $this->ticketTable t
                JOIN department dp ON t.department_id = dp.id
                JOIN users ur ON t.user_id = ur.id
                  WHERE  t.user_id=:userid AND  (t.title like :search or ur.name like :search or t.details like :search or dp.name like :search)";

           $stmt = $this->conn->prepare($sql);
           $stmt->bindParam(':userid',$this->userId);

        }

        $searchParam = '%' . $search . '%';
        $stmt->bindParam(':search',$searchParam);
        $stmt->execute();
        $list = [];

        while ($row = $stmt->fetch(PDO::FETCH_ASSOC))   $list[] = $row;
        return (json_encode( ['data' => $list], JSON_PRETTY_PRINT));




    }

}
```

Reply.php

```
    <?php

class Reply extends Database {
    private $replyTable;
    private $userId;


    public function __construct () {
        parent::__construct();
        $this->replyTable='replies';
      if (isset($_SESSION['id']))  $this->userId=$_SESSION['id'];

    }

    function saveReply($ticketId)   {
```

```
        if (isset($_POST['text_to_send'])) {

            $sql = "INSERT INTO replies (msg, ticket_id, user_id) VALUES (:msg,:ticket_id,:user_id)";
            $stmt = $this->conn->prepare($sql);
            $stmt->bindParam(':msg', $_POST['text_to_send']);
            $stmt->bindParam(':ticket_id', $ticketId);
            $stmt->bindParam(':user_id', $this->userId);
            $stmt->execute();

        }
    }


    function getReply($ticketId) {
        $sql = "SELECT r.id as reply_id, r.time_created as time_created, r.msg as msg, u.name as username
                FROM $this->replyTable r
                JOIN users u ON r.user_id = u.id
                JOIN tickets t ON r.ticket_id = t.id
                WHERE r.ticket_id = $ticketId
                ORDER BY time_created ASC";

        $data = $this->conn->query($sql);
        $list = $data->fetchAll(PDO::FETCH_ASSOC);
        return json_encode(['data' => $list], JSON_PRETTY_PRINT);

    }

}
```

Javascript

Reply.js

```
function printReplyData(item){
    var html = "<div class='border rounded bg-subtle m-3 pt-3 '>";
    html += item.msg ;
     html += "<div class='border rounded-bottom bg-secondary-subtle mt-3e '>" +"<i class='bi bi-person-circle'></i> " + item.username  ;
    html +=' <i class="bi bi-clock-history"></i> ' + item.time_created +"</div>"
    html += "</div>";
    $("#container").append(html);

}


$(document).ready(function () {
    var initialButtonValue = $("#replybutton").val();
```

```
    $.ajax({
        type: "POST",
        url: "reply_actions.php",
        data: { 'reply': initialButtonValue, },
        dataType: 'json',
        success: function (response) {

            console.log(response.data);
            $("#container").empty();
            $.each(response.data, function (index, item) {
            printReplyData(item);
            $('#replytext').val('');
            });
        },
    });



    $("#replybutton").click(function (e) {
        e.preventDefault();
        var buttonValue = $(this).val();
        var textareaValue = $("#replytext").val();

        $.ajax({
            type: "POST",
            url: "reply_actions.php",
            data: {
                'send_reply': buttonValue,
                'text_to_send': textareaValue
            },
            dataType: 'json',
            success: function (response) {

                console.log(response);
                $("#container").empty();
                $.each(response.data, function (index, item) {
                    printReplyData(item);
                    $('#replytext').val('');
                });
            },
        });
    });
});
```

Ticket.js

```
function printTicketData(item){
        var newRow = $('<tr></tr>');
        newRow.append('<td>' + item.ticket_id + '</td>');
```

```
        newRow.append('<td>' + item.ticket_title + '</td>');
            newRow.append('<td><div class="detailCellSize overflow-auto">' + item.ticket_details + '</div></td>');
        newRow.append('<td>' + item.department_name + '</td>');

            newRow.append(item.ticket_solved == 0 ? '<td class="text-success"><i class="bi bi-door-open"></i>Ανοιχτό</td>' : '<td class="text-danger"><i class="bi bi-door-closed"></i>Κλειστό</td>');
        newRow.append('<td><i class="bi bi-person-circle"></i> ' + item.user_name + '</td>');
                    newRow.append('<td><form action="./forms/edit_ticket.php" method="post"><button name="edit_ticket" value="' + item.ticket_id + '" type="submit" class="btn btn-outline-primary m-1"><i class="bi bi-pencil-square"></i>  </button></form></td>');
        newRow.append('<td><i class="bi bi-clock-history"></i> ' + item.time_created + '</td>');
          newRow.append('<td><a class="nav-link text-primary" id="replylink" href="./reply.php?reply=' + item.ticket_id + '"><i class="bi bi-chat"></i>     </a></td>');
        $('#ticketTable').append(newRow);
}


function loadTicketData(pageNr) {

        $.ajax({
        type: 'GET',
        url: 'ticket_actions.php',
        data: { 'page-nr': pageNr },
        dataType: 'json',

        success: function (response) {
           $('#ticketTable tbody').find('tr:gt(0)').remove();
           $.each(response.data, function (index, item) {
           printTicketData(item);

           });
        },
    });
}

    loadTicketData('1');









    $('.page-link').on('click', function (e) {
    e.preventDefault();
    var pageNr = $(this).data('page-nr');
    loadTicketData(pageNr);

    });

    $('#searchForm').submit(function (event) {
```

```
        event.preventDefault();
        var searchData = $("#search").val();

        $.ajax({
            type: 'POST',
            url: 'ticket_actions.php',
            data: { 'search': searchData, },
            dataType: 'json',

            success: function (response) {

                $('#ticketTable tbody').find('tr:gt(0)').remove();
                $.each(response.data, function (index, item) {
                printTicketData(item);

              });

            },
        });

    });



    $(document).ready(function(){
        $("#modalButton").click(function(){
          alert("Το ticket δημιουργήθηκε με επιτυχία!");
        });
      });




$('.dropdown-item').click(function (event) {

    event.preventDefault();
    var sortValue= $(this).attr('href');

if(sortValue==="order=ASC&collumn=ticket_solved")

{
    var filterVal = "ASC";
    var collumnVal = "ticket_solved";

}
else if(sortValue==="order=ASC&collumn=time_created"){
    var filterVal = "ASC";
    var collumnVal = "time_created";

}

else {
    var filterVal = "DESC";
```

```
    var collumnVal = "time_created";


}
    $.ajax({
        type: 'GET',
        url: 'ticket_actions.php',
        data: {
            'order': filterVal,
            'collumn': collumnVal
        },
        dataType: 'json',
        success: function (response) {
          console.log(response);
      $('#ticketTable tbody').find('tr:gt(0)').remove();
            $.each(response.data, function (index, item) {
                printTicketData(item);




            });









        },
    });



});
```

Βιβλιογραφία

[1] S. Pargaonkar, ‘A Comprehensive Research Analysis of Software Development Life Cycle (SDLC) Agile & Waterfall Model Advantages, Disadvantages, and Application Suitability in Software Quality Engineering’, *Int. J. Sci. Res. Publ.*, vol. 13, no. 8, pp. 120–124, Aug. 2023, doi: 10.29322/IJSRP.13.08.2023.p14015.

[2] ‘Frontend vs Backend’, GeeksforGeeks. Accessed: Jan. 23, 2024. [Online]. Available: https://www.geeksforgeeks.org/frontend-vs-backend/

[3] ‘HTML History’. Accessed: Jan. 12, 2024. [Online]. Available: https://www.w3schools.in/html/history

[4] ‘HTML: HyperText Markup Language | MDN’. Accessed: Jan. 12, 2024. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTML

[5] J. Krause, *Introducing Web Development*. Berkeley, CA: Apress, 2016. doi: 10.1007/978-1-4842-2499-1.

[6] ‘How does a web browser interpret HTML? | TutorChase’. Accessed: Jan. 13, 2024. [Online]. Available: https://www.tutorchase.com/answers/a-level/computer-science/how-does-a-web-browser-interpret-html

[7] ‘Cascading Style Sheets, designing for the Web – Chapter 20: The CSS saga’. Accessed: Jan. 17, 2024. [Online]. Available: https://www.w3.org/Style/LieBos2e/history/

[8] C. J. Wells, ‘A Brief History of CSS’. Accessed: Jan. 18, 2024. [Online]. Available: https://www.technologyuk.net/website-development/introduction-to-css/introduction.shtml

[9] ‘Cascading Style Sheets, designing for the Web – Chapter 2: CSS’. Accessed: Jan. 18, 2024. [Online]. Available: https://www.w3.org/Style/LieBos2e/enter/

[10] T. A. Powell and T. A. Powell, *HTML & CSS: the complete reference*, 5th ed. New York: McGraw-Hill, 2010.

[11] ‘Bootstrap Get Started’. Accessed: Jan. 22, 2024. [Online]. Available: https://www.w3schools.com/bootstrap/bootstrap_get_started.asp

[12] ‘A Brief History of JavaScript’, DEV Community. Accessed: Jan. 24, 2024. [Online]. Available: https://dev.to/dboatengx/history-of-javascript-how-it-all-began-92a

[13] ‘Introduction to JavaScript’, GeeksforGeeks. Accessed: Jan. 26, 2024. [Online]. Available: https://www.geeksforgeeks.org/introduction-to-javascript/

[14] ‘jQuery | Introduction’, GeeksforGeeks. Accessed: Jan. 29, 2024. [Online]. Available: https://www.geeksforgeeks.org/jquery-introduction/
[15] ‘AJAX Tutorial’. Accessed: Feb. 02, 2024. [Online]. Available: https://www.tutorialspoint.com/ajax/index.htm
[16] ‘PHP: History of PHP - Manual’. Accessed: Feb. 02, 2024. [Online]. Available: https://www.php.net/manual/en/history.php.php
[17] R. Nixon, Learning PHP, MySQL, JavaScript, and CSS: a step-by-step guide to creating dynamic websites, 2. ed. Beijing Köln: O’Reilly, 2012.
[18] ‘PHP: Introduction - Manual’. Accessed: Feb. 08, 2024. [Online]. Available: https://www.php.net/manual/en/introduction.php
[19] ‘What is a database?’ Accessed: Feb. 15, 2024. [Online]. Available: https://www.oracle.com/database/what-is-database/
[20] ‘SQL - Overview’. Accessed: Feb. 15, 2024. [Online]. Available: https://www.tutorialspoint.com/sql/sql-overview.htm
[21] ‘MySQL - Introdution’, GeeksforGeeks. Accessed: Feb. 15, 2024. [Online]. Available: https://www.geeksforgeeks.org/mysql-introdution/
[22] ‘Introduction — phpMyAdmin 5.1.4 documentation’. Accessed: Feb. 19, 2024. [Online]. Available: https://docs.phpmyadmin.net/en/latest/intro.html
[23] ‘XAMPP Tutorial - javatpoint’, www.javatpoint.com. Accessed: Feb. 19, 2024. [Online]. Available: https://www.javatpoint.com/xampp
[24] ‘What is Apache - Javatpoint’, www.javatpoint.com. Accessed: Feb. 21, 2024. [Online]. Available: https://www.javatpoint.com/what-is-apache