Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΛΟΠΟΝΝΗΣΟΥ

Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ανάπτυξη και Σχεδιασμός Ηλεκτρονικής Πλατφόρμας Κράτησης Ραντεβού για Ιατρική επίσκεψη

Σπουδαστές

ΚΑΡΑΜΟΥΖΗΣ ΠΑΝΑΓΙΩΤΗΣ - Α.Μ. :

Επιβλέπων Καθηγητής

Γεώργιος Ασημακόπουλος

ΠΑΤΡΑ 2021

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή

Πάτρα, Ημερομηνία

ΕΠΙΤΡΟΠΗ ΑΞΙΟΛΟΓΗΣΗΣ

1. Ονοματεπώνυμο, Υπογραφή

2. Ονοματεπώνυμο, Υπογραφή

3. Ονοματεπώνυμο, Υπογραφή

Υπεύθυνη Δήλωση Φοιτητή

Βεβαιώνω ότι είμαι συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης έχω αναφέρει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επίσης βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά ειδικά για τη συγκεκριμένη εργασία. Η έγκριση της διπλωματικής εργασίας από το Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Πανεπιστημίου Πελοποννήσου δεν υποδηλώνει απαραιτήτως και αποδοχή των απόψεων του συγγραφέα εκ μέρους του Τμήματος. Η παρούσα εργασία αποτελεί πνευματική ιδιοκτησία του φοιτητή Καραμούζης Παναγιώτης που την εκπόνησε. Στο πλαίσιο της πολιτικής ανοικτής πρόσβασης ο συγγραφέας/δημιουργός εκχωρεί στο Πανεπιστήμιο Πελοποννήσου, μη αποκλειστική άδεια χρήσης του δικαιώματος αναπαραγωγής, προσαρμογής, δημόσιου δανεισμού, παρουσίασης στο κοινό και ψηφιακής διάχυσής τους διεθνώς, σε ηλεκτρονική μορφή και σε οποιοδήποτε μέσο, για διδακτικούς και ερευνητικούς σκοπούς, άνευ ανταλλάγματος και για όλο το χρόνο διάρκειας των δικαιωμάτων πνευματικής ιδιοκτησίας. Η ανοικτή πρόσβαση στο πλήρες κείμενο για μελέτη και ανάγνωση δεν σημαίνει καθ' οιονδήποτε τρόπο παραχώρηση δικαιωμάτων διανοητικής ιδιοκτησίας του συγγραφέα/δημιουργού ούτε επιτρέπει την αναπαραγωγή, αναδημοσίευση, αντιγραφή, αποθήκευση, πώληση, εμπορική χρήση, μετάδοση, διανομή, έκδοση, εκτέλεση, «μεταφόρτωση» (downloading), «ανάρτηση» (uploading), μετάφραση, τροποποίηση με οποιονδήποτε τρόπο, τμηματικά ή περιληπτικά της εργασίας, χωρίς τη ρητή προηγούμενη έγγραφη συναίνεση του συγγραφέα/δημιουργού. Ο συγγραφέας/δημιουργός διατηρεί το σύνολο των ηθικών και περιουσιακών του δικαιωμάτων.

ΠΕΡΙΛΗΨΗ

Η πτυχιακή εργασία με τίτλο «Ανάπτυξη και Σχεδιασμός Ηλεκτρονικής Πλατφόρμας Κράτησης Ραντεβού για Ιατρική επίσκεψη» έχει ως κύριο σκοπό την δημιουργία μιας διαδικτυακής πλατφόρμας μέσω της οποίας θα κατανοήσουμε τον τρόπο με τον οποίο αλληλοεπιδρούν οι γλώσσες προγραμματισμού του διαδικτύου μεταξύ τους καθώς και τον τρόπο με τον οποίο συνδέονται με την βάση δεδομένων.

Η πλατφόρμα υλοποιείται με την χρήση των γλωσσών HTML, CSS, JavaScript, Bootstrap Framework και PHP. Έχει σχεδιαστεί για να χρησιμοποιείται από τρία είδη χρηστών τον διαχειριστή, τον ιατρό και τον ασθενή. Κάθε ρόλος χρήστη αποτελεί ξεχωριστή οντότητα στο σύστημα με τα δικά του χαρακτηριστικά, τις δικές του μεθόδους και λειτουργίες καθώς και τον δικό του πίνακα στην βάση δεδομένων.

Το πρώτο κεφάλαιο αναφέρεται στην γλώσσα προγραμματισμού και κατασκευής των βάσεων δεδομένων SQL. Παρουσιάζεται ο τρόπος σύνδεσης της βάσης του συστήματος με τα αρχεία κώδικα του project.

Στο δεύτερο κεφάλαιο αναλύεται λεπτομερώς η χρήση της PHP και ο τρόπος δόμησης του κώδικα ενώ στο τρίτο κεφάλαιο παρουσιάζεται ο διαμεσολαβητικός ρόλος της JavaScript ανάμεσα στην PHP-SQL και την HTML.

Στα επόμενα δύο κεφάλαια θα αναφερθούμε στην δημιουργία των αρχείων μορφοποίησης CSS και της επικοινωνίας τους με το Framework Bootstrap που χρησιμοποιείται για μια πιο προσεγμένη εμφάνιση της ιστοσελίδας σε επίπεδο χρήστη.

Τέλος στο έβδομο κεφάλαιο αναλύουμε την μεθοδολογία που ακολουθήσαμε για την ανάπτυξη του συστήματος, τον καταμερισμό των αρχείων και την οργάνωση των διαφορετικών οντοτήτων στην βάση δεδομένων.

Στο τελευταίο κεφάλαιο παρουσιάζονται τα συμπεράσματα στα οποία καταλήξαμε μέσα από την ερεύνα και την καταγραφή των τεχνολογιών στην παρούσα πτυχιακή.

Λέξεις Κλειδιά:

Διαδικτυακή πλατφόρμα, εφαρμογή, HTML, CSS, JavaScript, PHP, Bootstrap, Framework, ιστοσελίδα, ραντεβού, γλώσσες προγραμματισμού

ABSTRACT

The main goal of our thesis with the title "Development and Design of an Online Medical Appointment Booking Platform" is to create an online platform in order to understand the way that web programming languages interact with each other and with the linked database.

The platform is implemented using HTML, CSS, JavaScript, Bootstrap Framework and PHP. It is designed to be used by three different types of users: the administrator, the doctor and the patient. Each user role is a separate entity in the system with its own characteristics, methods and functions as well as its own table in the database.

The first chapter deals with the programming language SQL which is used for the construction of the databases. It is also presented the way to connect the system database with the project code files.

The second chapter discusses in detail the use of PHP and how the code is structured, while the third chapter presents the mediating role of JavaScript between PHP-SQL and HTML.

In the next two chapters we will refer to the creation of CSS formatting files and their communication with the Framework Bootstrap that is used for a more careful appearance of the website at the user level.

Finally in the seventh chapter we analyze the methodology we followed for the development of the system, the sharing of files and the organization of the different entities in the database.

The last chapter presents the conclusions that we came to through the research and recording of technologies in our thesis.

Keywords:

Web platform, application, HTML, CSS, JavaScript, PHP, Bootstrap, Framework, website, appointment, programming languages

Ευχαριστώ ιδιαίτερα τον επιβλέποντα καθηγητή μου κο. Ασημακόπουλο Γεώργιο για την εμπιστοσύνη που μου έδειξε στην ανάληψη της εργασίας καθώς και την οικογένεια μου που με στήριξε όλα αυτά τα χρόνια των σπουδών μου.

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΙΝΑΚΕΣ

ΓΡΑΦΗΜΑΤΑ – ΕΙΚΟΝΕΣ

1. ΕΙΣΑΓΩΓΗ

Στην παρούσα πτυχιακή θα αναφερθούμε στις βασικές ιδιότητες των γλωσσών προγραμματισμού HTML, CSS, JavaScript, PHP καθώς και το Framework Bootstrap. Θα παρουσιάσουμε τον τρόπο σύνταξης των αρχείων της κάθε γλώσσας αλλά και τον τρόπο σύνδεσης μεταξύ των διαφορετικών αρχείων ώστε να μπορούν να συνυπάρχουν και να επιτελούν τις λειτουργίες που έχουμε ορίσει.

Επιπρόσθετα θα παρουσιάσουμε τον τρόπο με τον οποίο κατασκευάσαμε την βάση δεδομένων της πλατφόρμας μας, πως την συνδέσαμε με τα αρχεία του κώδικα και πως καταφέραμε να αντλούμε ή να εισάγουμε νέα δεδομένα στην βάση.

Θα πρέπει να αναφέρουμε ότι στο τελευταίο κεφάλαιο αναλύεται λεπτομερώς η μεθοδολογία που ακολουθήσαμε για να δημιουργήσουμε την ιστοσελίδα μας καθώς και η λειτουργία και ο λόγος ύπαρξης του κάθε αρχείου ξεχωριστά.

Επηρεασμένοι μετά από δύο χρόνια παγκόσμιας κρίσης λόγω της ασθένειας Covid-19 αποφασίσαμε να κατασκευάσουμε μια πλατφόρμα ιατρικής φύσεως που θα μπορεί κάποιος χρήστης να εγγραφεί και να κλείσει άμεσα ραντεβού με οποιοδήποτε ιατρό έχει καταχωρηθεί στην βάση δεδομένων από τον διαχειριστή της υποτιθέμενης κλινικής. Τα στοιχεία των ιατρών και της κλινικής που αναφέρονται στην πλατφόρμα είναι προϊόν φαντασίας και εξυπηρετούν καλύτερα την χρήση και την προβολή της λειτουργικότητας της ιστοσελίδας.

Η πλατφόρμα Medical που κατασκευάσαμε περιλαμβάνει τρεις διαφορετικούς ρόλους χρήστη, τον διαχειριστή, τον ασθενή και τον γιατρό. Οι τρείς αυτοί διαφορετικοί χρήστες έχουν την δυνατότητα να συνδεθούν στην πλατφόρμα, να κλείσουν κάποιο ραντεβού, να μεταβάλλουν τα προσωπικά στοιχεία τους στην βάση δεδομένων, να εγγραφούν εκ νέου στην βάση και να διαγράψουν ή να αλλάξουν το ωράριο εργασίας τους ή κάποιο ήδη κλεισμένο ραντεβού.

Για την προβολή της κεντρικής σελίδας της πλατφόρμας αρκεί να εισάγετε το παρακάτω URL: http://localhost/medical/index.php. Πατώντας στο κουμπί «ΣΥΝΔΕΣΗ» θα μεταβείτε στην σελίδα της σύνδεσης με την αντίστοιχη φόρμα για να εισάγετε τους κωδικούς του κάθε χρήστη. Για την εύκολη πρόσβαση λόγω εκπαιδευτικής δραστηριότητας έχουμε τοποθετήσει έναν πίνακα με τους κωδικούς ενδεικτικών χρηστών που υπάρχουν καταχωρημένοι στην βάση δεδομένων ώστε να μπορέσετε πιο εύκολα να μεταβείτε στο dashboard του χρήστη.

Η βάση δεδομένων έχει δημιουργηθεί με την βοήθεια του Framework PhpMyAdmin σε XAMPP.

Η παρούσα πτυχιακή συνοδεύεται με τον φάκελο των αρχείων που κατασκευάστηκαν για την πλατφόρμα αλλά και το αρχείο της ίδιας της βάσης δεδομένων medical.

2. Η ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ ΒΑΣΕΩΝ ΔΕΔΟΜΕΝΩΝ SQL

Στο πρώτο κεφάλαιο θα ασχοληθούμε με την παρουσίαση και την επεξήγηση των βασικών στοιχείων και παραμέτρων της γλώσσας προγραμματισμού SQL. Ειδικότερα θα δοθούν οι βασικές έννοιες της γλώσσας, ο τρόπος χρήσης της και σύνδεσης της με τις υπόλοιπες γλώσσες προγραμματισμού που απαιτούνται για την ανάπτυξη ενός ηλεκτρονικού συστήματος, καθώς και ο τρόπος εξαγωγής καίριων αποτελεσμάτων από την βάση δεδομένων μέσω της μορφής των ερωτημάτων.

2.1.Βασικές Αρχές της SQL

Ξεκινώντας με την παρουσίαση των βασικών στοιχείων της γλώσσας θα πρέπει πρώτα να αναφερθούμε στο ιστορικό υπόβαθρο που οδήγησε στην δημιουργία της.

Η ανάπτυξη της SQL ξεκίνησε την δεκαετία του 1970 από την IBM με τους εμπνευστές και δημιουργούς της να είναι οι Andrew Richardson, Donald C. Messerly και Raymond F. Boyce. Σκοπός της ανάπτυξης μιας τέτοιας γλώσσας ήταν ο χειρισμός και η ανάκτηση στοιχείων που είχαν αποθηκευτεί στο πρώτο RDBMS σύστημα της IBM ή αλλιώς σύστημα R.

Το σύστημα R αποτελεί το πρώτο σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων (RDBMS) που αναπτύχθηκε στις αρχές της δεκαετίας του 1970 από το πανεπιστήμιο του MIT έχοντας ως αντίπαλο του την ίδια περίοδο το σύστημα Ingres το οποίο αναπτύχθηκε το 1974 από το πανεπιστήμιο του Berkley. Αρχικά το σύστημα Ingres σχεδιάστηκε με σκοπό να εφαρμόσει την γλώσσα διατύπωσης ερωτήσεων QUEL η οποία όμως με την πάροδο των ετών και την ανάπτυξη επιμέρων τμημάτων στον τομέα των σχεσιακών συστημάτων αντικαταστάθηκε στην αγορά από την SQL.

Η SQL (Structured Query Language) ορίζεται ως μια γλώσσα προγραμματισμού για την κατασκευή βάσεων δεδομένων, που σχεδιάστηκε για την διαχείριση δεδομένων, σε ένα σύστημα διαχείρισης σχεσιακών βάσεων (Relational Database Management System, RDBMS).

Αξίζει να σημειωθεί ότι το αρχικό προσωνύμιο που δόθηκε στην γλώσσα δεν ήταν SQL αλλά SEQUEL και η ανάπτυξη της βασίστηκε εξαρχής στην σχεσιακή άλγεβρα, σχεδιασμένη από τον

Edgar F. Codd το 1970. Προς το τέλος της δεκαετίας του 1970 η τότε άγνωστη εταιρεία Relational Software γνωστή σήμερα ως Oracle Corporation διακρίνοντας τις δυνατότητες των σχεσιακών συστημάτων διαχείρισης δεδομένων που περιεγράφηκαν από τους Codd, Chamberlin, και Boyce, αναλαμβάνει να φέρει εις πέρας το όραμα τους δημιουργώντας την σημερινή μορφή της SQL φιλοδοξώντας στην πώληση της στο Αμερικάνικο Ναυτικό, την Κεντρική Υπηρεσία Πληροφοριών και σε άλλες επιμέρους Αμερικανικές Υπηρεσίες.

Η σημαντικότερη εξέλιξη στην αγορά των σχεσιακών συστημάτων διαχείρισης δεδομένων έρχεται να λάβει χώρα το καλοκαίρι του 1979 όταν η Relational Software εισάγει την πρώτη διαθέσιμη εφαρμογή της SQL στο εμπόριο καταφέρνοντας έτσι να ξεπεράσει την IBM κατακτώντας για πρώτη φορά την αγορά ένα RDBMS. Η εφαρμογή SQL αποδεικνύει την αξία της και την νίκη της έναντι άλλων συναφών προϊόντων τεχνολογίας παραμένοντας στις πρώτες θέσεις των τάσεων της αγοράς για αρκετές εβδομάδες μετά την πρώτη κυκλοφορία της.

Σήμερα τα πιο συχνά προς χρήση σχεσιακά συστήματα διαχείρισης που χρησιμοποιούν SQL είναι η Oracle, Sybase, Microsoft SQL Server, Access, Ingres, κ.α.. Παρότι διαφορετικά συστήματα χρησιμοποιούν κοινή γλώσσα για την διαχείριση των δεδομένων τους αξίζει να σημειωθεί ότι τα περισσότερα έχουν δημιουργήσει μια πληθώρα νέων πρόσθετων και εξειδικευμένων ιδιόκτητων επεκτάσεων που καλούνται να συνεργαστούν με την SQL για την καλύτερη ανάκτηση και οργάνωση των δεδομένων. Το γεγονός όμως αυτό δεν μπορεί να αναιρέσει την αξία και τις δυνατότητες της ίδιας της γλώσσας, καθώς η SQL μέσω των εντολών της μπορεί αυτεξούσια να εκτελέσει οποιαδήποτε ενέργεια απαιτείται για την διαχείριση του συστήματος χωρίς την επιπλέον βοήθεια των πρόσθετων επεκτάσεων.

Το ερώτημα όμως παραμένει: τι μπορεί στην ουσία να κάνει η SQL;

Οι βασικές λειτουργίες της γλώσσας που καλείται να εκτελέσει πάνω σε μια βάση δεδομένων είναι οι εξής:

1. Εκτέλεση ερωτημάτων,
2. Ανάκτηση δεδομένων,
3. Εισαγωγή νέων εγγραφών ή στοιχείων,
4. Ενημέρωση των ήδη υπαρχόντων εγγραφών,
5. Διαγραφή ολόκληρων εγγραφών ή στοιχείων,

6. Δημιουργία νέας βάσης δεδομένων,
7. Δημιουργία νέων πινάκων στην βάση δεδομένων,
8. Δημιουργία και αποθήκευση διεργασιών στην βάση για μετέπειτα εκτέλεση τους,
9. Δημιουργία προβολών στην βάση δεδομένων και
10. Καθορισμός δικαιωμάτων χρήσης σε πίνακες, διαδικασίες και προβολές της βάσης.

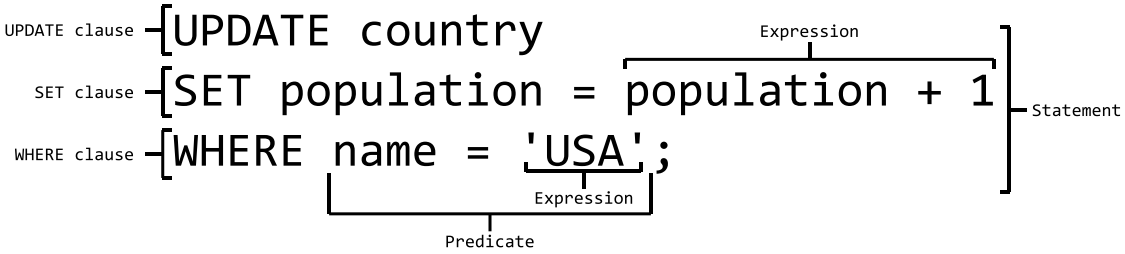
Οι λειτουργίες όμως αυτές χρειάζονται συγκεκριμένες εντολές για να εκτελεστούν. Έτσι έχουμε τις παρακάτω εντολές των οποίων ο τρόπος λειτουργίας θα παρουσιασθεί στο παρακάτω κεφάλαιο *(Βλέπε Κεφάλαιο 2.2):*

- **Select,**
- **Insert,**
- **Update,**
- **Delete,**
- **Create**
- **και Drop**

Ένα ακόμη σημαντικό χαρακτηριστικό της γλώσσας που αξίζει να σημειωθεί είναι η πολυπληθής υποδιαίρεση της σε επιμέρους γλωσσικά στοιχεία των οποίων τα ονόματα και ο τρόπος λειτουργίας τους παρουσιάζεται στην παρακάτω λίστα:

- **Clauses:** Είναι ένα προαιρετικό στοιχείο που όμως είναι απαραίτητο συστατικό για την δήλωση των ερωτημάτων
- **Expressions:** Είναι το στοιχείο που χρησιμοποιείται για την παραγωγή κλιμακωτών τιμών και την προβολή πινάκων που αποτελούνται από στήλες και σειρές στοιχείων
- **Predicates:** Αναλαμβάνουν την αποσαφήνιση και ακριβή διατύπωση των όρων που μπορούν να απαντηθούν με σωστό ή λάθος
- **Queries:** Αποτελούν τα ερωτήματα που διατυπώνονται με σκοπό την ανάκτηση συγκεκριμένων στοιχείων από την βάση δεδομένων

- **Statements:** Είναι υπεύθυνα για την ροή, την δημιουργία και την σύνδεση του προγράμματος με άλλα συναφή προγράμματα
- **Κενό:** Είναι ένα στοιχείο που κυρίως αγνοείται κατά την διάρκεια της ροής του προγράμματος από τα υπόλοιπα γλωσσικά στοιχεία. Η χρήση του όμως απαιτείται για τον διαχωρισμό μεταξύ των διαφορετικών συστατικών των Statements.



Εικόνα 1 Παράδειγμα συνόλου των γλωσσικών στοιχείων της SQL

2.2. Δημιουργία βάσης δεδομένων και πινάκων

Όπως αναφέραμε και στο προηγούμενο κεφάλαιο εδώ θα παρουσιάσουμε τον τρόπο λειτουργίας των κύριων εντολών της SQL που χρησιμοποιούνται για την δημιουργία της ίδιας της βάσης δεδομένων καθώς και για την διαχείριση, κατανομή και ανάκτηση των δεδομένων της σε πίνακες, οι οποίοι αποτελούν τον βασικότερο στοιχείο μιας βάσης δεδομένων.

Συνοπτικά λοιπόν έχουμε τα εξής:

Εντολή Select: Χρησιμοποιείται για την επιλογή στοιχείων ή εγγραφών από την βάση δεδομένων.

Εντολή Insert: Απαιτείται για την λειτουργία της εισαγωγής νέων εγγραφών και στοιχείων στους πίνακες της βάσης δεδομένων

Εντολή Update: Είναι υπεύθυνη για την ενημέρωση των πινάκων και της ίδιας της βάσης δεδομένων όταν επιθυμούμε την αλλαγή κάποιων επιμέρους στοιχείων της βάσης

Εντολή Delete: Χρησιμοποιείται για την διαγραφή εγγραφών και στοιχείων εγγραφής

Εντολή Create: Είναι το αντίπαλο δέος της delete και είναι υπεύθυνη για την δημιουργία νέων εγγραφών, πινάκων ή εξ' ολοκλήρου νέας βάσης

Εντολή Drop: Αναλαμβάνει την διαγραφή πινάκων ή σχέσεων μεταξύ πινάκων ή της ίδιας της βάσης δεδομένων

Σημείωση! Παρατηρείται ότι οι εντολές delete και drop επιτελούν την ίδια εργασία όμως πρέπει να προσέξουμε πολύ γιατί υπάρχουν σημαντικές διαφορές στην λειτουργία τους που μας βοηθούν να τις ξεχωρίσουμε. Ας δούμε όμως κάποιες από αυτές τις σημαντικές διαφορές:

Στοιχείο σύγκρισης	**DELETE**	**DROP**
Σκοπός	Διαγραφή εγγραφών ή στοιχείων από πίνακα	Διαγραφή πινάκων, σχέσεις μεταξύ πινάκων ή βάση δεδομένων
Γλώσσα	Είναι εντολή DML	Είναι εντολή DDL
Clause ή Ρήτρα	Χρησιμοποιείται όταν υπάρχει ρήτρα φιλτραρίσματος	Χρησιμοποιείται όταν δεν υπάρχει ρήτρα
Rollback ή Επιστροφή	Η εντολή delete μπορεί να αναιρεθεί αφού λειτουργεί σε buffer δεδομένων	Η εντολή drop δεν μπορεί να αναιρεθεί γιατί δουλεύει απευθείας πάνω στα δεδομένα
Χωρητικότητα μνήμης	Η χωρητικότητα μνήμης του πίνακα δεν απελευθερώνεται ακόμη και αν διαγραφούν όλες οι εγγραφές	Η εντολή απελευθερώνει την χωρητικότητα μνήμης
Πρόβλημα	Η χρήση της εντολής μπορεί να αντιμετωπίσει πρόβλημα έλλειψης χωρητικότητας μνήμης	Η χρήση της εντολής μπορεί να προκαλέσει αδυναμία στον κατακερματισμό της μνήμης

Interaction ή Αλληλεπίδραση	Χρησιμοποιείται η γλώσσα SQL που αλληλοεπιδρά σε απευθείας σύνδεση με τον server	Χρησιμοποιείται η γλώσσα PL/SQL που δεν επικοινωνεί απευθείας με τον διακομιστή της βάσης δεδομένων
Orientation ή Προσανατολισμός	Η SQL είναι γλώσσα προσανατολισμένη στα δεδομένα (data-oriented)	Η PL/SQL είναι γλώσσα προσανατολισμένη στην εφαρμογή (application oriented)
Αντικείμενο	Η SQL χρησιμοποιείται για την σύνταξη ερωτημάτων και την δημιουργία και εκτέλεση καταστάσεων DML και DDL	Η PL/SQL χρησιμοποιείται για την σύνταξη μπλοκ προγράμματος, συναρτήσεων, διαδικασιών, triggers και πακέτων.

Πίνακας 1 Διαφορές μεταξύ των εντολών Delete και Drop της γλώσσας διαχείρισης βάσεων δεδομένων SQL.

2.2.1. Τρόπος σύνταξης του αρχείου της βάσης δεδομένων SQL

Ο τρόπος σύνταξης στην γλώσσα SQL θα μπορούσε να χαρακτηριστεί αρκετά εύκολος, εύχρηστος και ευθύς σε σύγκριση πάντα με άλλες γλώσσες προγραμματισμού. Δεν απαιτείται η χρήση εισαγωγικών ή άλλων συμβόλων όταν αναφερόμαστε σε πίνακες, στήλες ή σειρές ενός πίνακα ή στην ίδια την βάση δεδομένων.

Η διαχείριση της βάσης δεδομένων μέσω κώδικα γίνεται με την μορφή των statements ή αλλιώς δηλώσεων. Η γενική μορφή ενός statement αποτελείται από τις κύριες εντολές ή αλλιώς keywords της SQL, τους identifiers – μεταβλητές που βοηθούν να αναγνωρίζουμε τους πίνακες, τα στοιχεία, τα αντικείμενα ή τις στήλες και σειρές των πινάκων μεταξύ τους και τα expressions – εκφράσεις που αποτελούν τις συνθήκες που θα ελεγχθούν για να δημιουργήσουν τα αποτελέσματα των ερωτημάτων.

Παρότι όπως αναφέραμε δεν απαιτούνται σύμβολα και εισαγωγικά για την ανάπτυξη των δηλώσεων, εντούτοις πρέπει να σημειωθεί ότι κάθε statement ακολουθείται στο τέλος από ένα

πολύ σημαντικό και απαραίτητο σύμβολο, το ελληνικό ερωτηματικό «;» το οποίο δηλώνει στην SQL το τέλος του συγκεκριμένου block κώδικα.

Έχοντας αυτά κατά νου παραθέτουμε το παρακάτω statement ως παράδειγμα καθώς και την επεξήγηση του:

```
SELECT * FROM Customers;
```

- Η εντολή select επιλέγει εγγραφές ή στοιχεία
- Η απόστροφος χρησιμοποιείται όταν θέλουμε να επιλέξουμε όλες τις εγγραφές ή στοιχεία
- Το στοιχείο from δηλώνει τον προορισμό από τον οποίο θέλουμε να εξάγουμε τις πληροφορίες και ο προορισμός αυτός στην δική μας περίπτωση είναι ο πίνακας customers

Έτσι με το συγκεκριμένο statement δηλώνουμε ότι θέλουμε να επιλεγούν όλες οι εγγραφές και τα στοιχεία του πίνακα customers.

Ακολουθώντας αυτήν την λογική παραθέτουμε το γενικό σύνολο των statements που μπορούμε να συντάξουμε με τη χρήση των κύριων εντολών της SQL, καθώς και την επεξήγηση της λειτουργίας τους.

2.2.1.1. Τα Statements της SQL

SELECT Statement:

Χρησιμοποιείται για την επιλογή στοιχείων από μια βάση δεδομένων. Συντάσσεται ως εξής όταν θέλουμε να επιλέξουμε στοιχεία από συγκεκριμένες στήλες ενός πίνακα:

```
SELECT column1, column2, ...
FROM table_name;
```

ή κατά αυτόν τον τρόπο όταν θέλουμε να επιλέξουμε όλα τα στοιχεία του πίνακα:

```
SELECT * FROM table_name;
```

SELECT DISTINCT Statement:

Χρησιμοποιείται όταν ξέρουμε ότι σε ένα πίνακα υπάρχουν διπλότυπες εγγραφές και εμείς θέλουμε να επιλέξουμε μόνο τα στοιχεία που δεν είναι διπλότυπα. Η σύνταξη είναι παρόμοια με την απλή select μόνο που εδώ χρειάζεται να προσθέσουμε το διακριτικό distinct:

```
SELECT DISTINCT column1, column2, ...
FROM table_name;
```

Στην παραπάνω δήλωση επιλέγουμε τις μη διπλότυπες εγγραφές που βρίσκονται στις στήλες column1 και column2 από τον πίνακα table_name.

SELECT INTO Statement:

Μια ακόμη παραλλαγή της εντολής select είναι το select into με την οποία μπορούμε να επιλέξουμε συγκεκριμένες εγγραφές ενός παλιού πίνακα, να τις αντιγράψουμε και έπειτα τα αντίγραφα να τα εισάγουμε σε κάποιο νέο πίνακα. Η εντολή αυτή συντάσσεται με αυτόν τον τρόπο:

```
SELECT *
INTO newtable [IN externaldb]
FROM oldtable
WHERE condition;
```

Με την παραπάνω εντολή επιλέγουμε με τον αστερίσκο (*) όλες τις εγγραφές του πίνακα oldtable στις οποίες ικανοποιείται μια συγκεκριμένη συνθήκη – condition, δημιουργούνται κρυφά αντίγραφα των εγγραφών και κατόπιν εισάγονται με την λέξη into στον πίνακα newtable που βρίσκεται - «είναι μέσα» - (λέξη in) στην βάση δεδομένων externaldb. Εάν θέλουμε να επιλέξουμε μόνο συγκεκριμένες στήλες τότε θα πρέπει να γράψουμε:

```
SELECT column1, column2, column3, ...
INTO newtable [IN externaldb]
FROM oldtable
WHERE condition;
```

Όπως βλέπουμε εδώ επιλέγουμε συγκεκριμένες στήλες αναγράφοντας δίπλα από την λέξη select τα ονόματα των στηλών διαχωριζόμενα μεταξύ τους με κόμμα.

INSERT INTO Statement:

Η χρήση του απαιτείται όταν θέλουμε να εισάγουμε στοιχεία - εγγραφές σε ένα πίνακα συντάσσεται με τον εξής τρόπο:

```
INSERT INTO table_name
VALUES (value1, value2, value3, ...);
```

Στην δήλωση αυτή εισάγουμε τα στοιχεία μιας εγγραφής στον πίνακα χωρίς να προσδιορίζουμε σε ποιες στήλες θα πρέπει να εισαχθεί το κάθε ένα στοιχείο, γι' αυτό θα πρέπει να προσέξουμε την σειρά με την οποία εισάγουμε τα στοιχεία ώστε να καταγραφούν με την αντίστοιχη σειρά στις σωστές στήλες. Εάν δεν γνωρίζουμε με ποια σειρά έχουν δημιουργηθεί οι στήλες στον πίνακα τότε μπορούμε να χρησιμοποιήσουμε την παρακάτω δήλωση στην οποία αναφέρουμε μετά το όνομα του πίνακα τα ονόματα των στηλών. Η λέξη values προσδιορίζει τις τιμές των στοιχείων που θα εισαχθούν στον πίνακα.

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

UPDATE Statement:

Χρησιμοποιείται για την αλλαγή στοιχείων – εγγραφών ενός πίνακα και συντάσσεται με την εντολή update και την λέξη set. Με την δήλωση αυτή αλλάζουμε στην στήλη name του πίνακα customers τις τιμές όλων των εγγραφών με την τιμή "lucas".

```
UPDATE Customers
SET Name='Lucas';
```

Αν δεν θέλουμε να αλλάξουμε τις τιμές όλων των εγγραφών στην στήλη name αλλά μόνο κάποια/ες τότε χρησιμοποιούμε την παρακάτω δήλωση:

```
UPDATE Customers
SET Name='Lucas'
WHERE Country='Mexico';
```

Με την δήλωση αυτή επιλέγουμε μόνο τις εγγραφές όπου στην στήλη country έχουν την τιμή "mexico" και κατόπιν αλλάζουμε την τιμή που έχουν στην στήλη name σε "lucas". Την διακριτική λέξη where θα την αναλύσουμε ενδελεχώς παρακάτω.

DELETE Statement:

Το statement αυτό θα το χρειαστούμε αν θέλουμε να διαγράψουμε στοιχεία ή εγγραφές κάποιου πίνακα και συντάσσεται:

```
DELETE FROM table_name;
```

Με αυτόν τον τρόπο διαγράφουμε όλα τα στοιχεία – εγγραφές του εκάστοτε πίνακα αν θέλουμε να διαγράψουμε κάποιες συγκεκριμένες εγγραφές τότε γράφουμε:

```
DELETE FROM table_name WHERE condition;
```

Με τον παραπάνω κώδικα δηλώνουμε την διαγραφή των εγγραφών που τα στοιχεία τους πληρούν κάποιες προϋποθέσεις και εγκρίνουν μια συνθήκη.

Μετά την καταγραφή των κυριότερων statements της SQL θα πρέπει να ασχοληθούμε και να αναφέρουμε επίσης τα σημαντικότερα clauses και operators που μπορούμε να χρησιμοποιήσουμε ώστε να αυξήσουμε τις δυνατότητες των παραπάνω statements. Έτσι λοιπόν παρουσιάζουμε τα παρακάτω clauses και operators με την σύνταξη και την επεξήγηση τους.

2.2.1.2. Τα Clauses της SQL

WHERE Clause:

Η λέξη where χρησιμοποιείται σε όλα τα statements για την δήλωση μιας συνθήκης δηλαδή ακολουθείται πάντα από μια συνθήκη που θα πρέπει να ικανοποιηθεί ώστε το statement να παράγει τα σωστά αποτελέσματα. Η σύνταξη όπως φαίνεται είναι απλή:

```
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

Στο παρακάτω παράδειγμα βλέπουμε ότι επιλέγουμε όλες τις εγγραφές του πίνακα customers όταν η τιμή στην στήλη τους country είναι “mexico”. Με λίγα λόγια επιλέγουμε όλους τους πελάτες που είναι από το Μεξικό.

```
SELECT * FROM Customers
WHERE Country='Mexico';
```

ORDER BY Clause:

Με το clause order by μπορούμε όταν το εισάγουμε σε οποιοδήποτε statement να ταξινομήσουμε τα παραγόμενα αποτελέσματα της δήλωσης κατά αύξουσα ή φθίνουσα σειρά.

Συντάσσεται ως εξής:

```
SELECT column1, column2, ...
FROM table_name
ORDER BY column1, column2, ... ASC|DESC;
```

Όπως παρατηρούμε στην παραπάνω δήλωση επιλέγουμε τις εγγραφές των στηλών column1 και column2 του πίνακα table_name και έπειτα τις εγγραφές αυτές τις ταξινομούμε βάσει των στηλών column1 και column2 σε αύξουσα σειρά ή φθίνουσα αναγράφοντας στο τέλος μία από τις δύο λέξεις ASC / DESC. Η λέξη ASC (ASCENDING) παράγει μια ταξινόμηση σε αύξουσα σειρά ενώ για φθίνουσα ταξινόμηση χρησιμοποιούμε την λέξη DESC (DESCENDING).

2.2.1.3. Οι Operators της SQL

Όπως αναφέραμε και παραπάνω η ύπαρξη και η χρήση της SQL βασίζεται πάνω στην σύνταξη ερωτημάτων τα οποία με την σειρά τους χρησιμοποιούν διαφόρων ειδών στοιχεία κώδικα ώστε να εκτελεστούν και να υπολογιστούν σωστά. Μερικά από αυτά τα στοιχεία μπορεί να είναι οι ρήτρες εκ των οποίων η πιο δημοφιλής είναι η WHERE, οι επονομαζόμενες προβλέψεις οι οποίες κατ' ουσίαν είναι οι συνθήκες που μπορούν να προσδιοριστούν από μια δυαδική τιμή, οι εκφράσεις δηλαδή αριθμητικά αποτελέσματα τελεστών ή συμβολοσειρές, τα ονόματα αντικειμένων όπως πίνακες, προβολές, στήλες, λειτουργίες και τέλος οι τιμές που μπορεί να είναι αριθμητικές ή συμβολοσειρές.

Όλα τα παραπάνω στοιχεία κώδικα που αναφέρθηκαν για να μπορέσουν να συγκρίνουν, να υπολογίσουν, να αποθηκεύσουν ή ακόμα και να εκλογικεύσουν μια συνθήκη χρησιμοποιούν τους τελεστές ή αλλιώς operators. Οι τελεστές μπορεί να είναι αριθμητικά σύμβολα ή λέξεις που αναλαμβάνουν να εκτελέσουν για παράδειγμα μια σύγκριση μεταξύ τιμών ή μια καταχώρηση ενός αποτελέσματος σε μία μεταβλητή.

Στον παρακάτω πίνακα παρουσιάζεται μια λίστα με τους βασικότερους τελεστές που χρησιμοποιούνται συχνότερα σε ένα κώδικα SQL καθώς και η λειτουργία που επιτελούν.

Τελεστής / Operator	**Λειτουργία**
=	Μια τιμή είναι ίση με κάποια άλλη
< > ή !=	Δύο τιμές είναι άνισες μεταξύ τους
>	Η πρώτη τιμή είναι μεγαλύτερη από την δεύτερη
<	Η πρώτη τιμή είναι μικρότερη από την δεύτερη

>=	Η πρώτη τιμή είναι μεγαλύτερη ή ίση από την δεύτερη
<=	Η πρώτη τιμή είναι μικρότερη ή ίση από την δεύτερη
BETWEEN	Δηλώνουμε ότι η τιμή που αναζητούμε είναι ανάμεσα σε ένα εύρος τιμών
LIKE	Η τιμή που θέλουμε ταιριάζει με ένα μοτίβο χαρακτήρων ή αριθμών
IN	Μια τιμή είναι ίση με μία ή περισσότερες τιμές
IS ή IS NOT	Ένα αποτέλεσμα ή μια μεταβλητή συγκρίνεται με το κενό – NULL
IS [NOT] TRUE ή IS [NOT] FALSE	Σύγκριση μεταξύ τιμών όπου το αποτέλεσμα της θα είναι δυαδικής μορφής ή αλλιώς boolean
IS NOT DISTINCT FROM	Συγκρίνονται δύο τιμές μεταξύ τους και ή θα είναι ίσες ή θα ισούνται με δύο μηδενικά
AS	Χρησιμοποιείται όταν θέλουμε να αλλάξουμε το όνομα μια στήλης ή ενός πίνακα κατά την διάρκεια προβολής των αποτελεσμάτων ενός ερωτήματος.

Πίνακας 2 Παρουσίαση των τελεστών της SQL και της λειτουργίας τους

2.2.1.4. Ενώσεις μεταξύ πινάκων και συσχετίσεις δεδομένων

Έχοντας αναφέρει τα βασικά στοιχεία που απαιτούνται για την σύνταξη και την εκτέλεση ενός κώδικα SQL ήρθε η ώρα να αναφέρουμε τον κυριότερο ρόλο που επιτελεί η γλώσσα SQL αυτόν της δημιουργίας πινάκων που αναλαμβάνουν να αποθηκεύσουν τον όγκο δεδομένων που έχουμε προκαθορίσει καθώς και την μεταξύ τους σύνδεση - ένωση που συντελεί στην παρουσίαση των δεδομένων μέσα από στοχευμένα ερωτήματα.

Η SQL στην βάση της είναι μια γλώσσα που χρησιμοποιείται κατά κόρον για την σύνδεση διαφορετικών αντικειμένων (πινάκων) με σκοπό την εξαγωγή συγκεκριμένων αποτελεσμάτων.

Βασίζεται στο δημοφιλέστερο μοντέλο καταχώρησης όγκου δεδομένων που είναι το σχεσιακό μοντέλο βάσης δεδομένων.

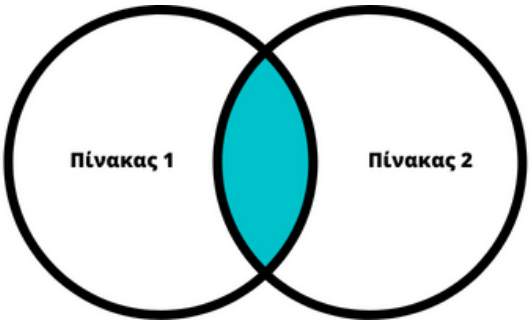
Σύνδεση ή αλλιώς ένωση μεταξύ πινάκων έχουμε όταν θέλουμε να εμφανίσουμε ως αποτέλεσμα ενός ερωτήματος ένα σύνολο από σειρές πολλαπλών πινάκων. Η σύνδεση βασίζεται πάντα σε ένα κοινό πεδίο που υπάρχει ανάμεσα στους δύο πίνακες. Το κοινό αυτό πεδίο συχνά είναι το πρωτεύων κλειδί των πινάκων. Πρωτεύων κλειδί είναι ένα πεδίο που θεωρείται στον πίνακα μοναδικό για παράδειγμα σε ένα πίνακα με τα στοιχεία των φοιτητών μιας σχολής πρωτεύων κλειδί θα είναι ο αριθμός μητρώου του κάθε φοιτητή.

Ο πιο κοινός τύπος ένωσης πινάκων είναι ο inner join (εσωτερική συνένωση). Ο τύπος αυτός επιστρέφει όλες τις γραμμές των δύο πινάκων που συνδέονται όταν φυσικά ικανοποιείται η συνθήκη που έχουμε προκαθορίσει στο ερώτημα προβολής της ένωσης.

Σκοπός της κάθε ένωσης είναι να εμφανιστούν όλες οι εγγραφές των πινάκων για τις οποίες επιβεβαιώνεται η συνθήκη χωρίς όμως να υπάρχουν διπλότυπες εγγραφές.

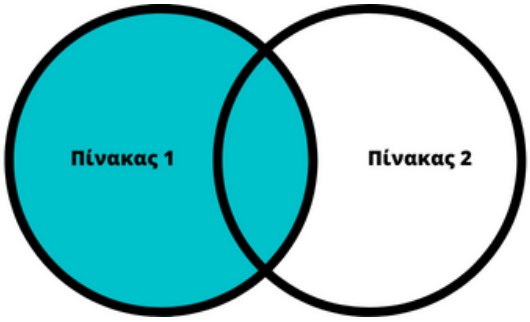
Εκτός από την συνένωση inner join έχουμε και την outer join η οποία διακρίνεται σε τρείς διαφορετικές προεκτάσεις. Έτσι έχουμε τους παρακάτω τύπους ένωσης:

- **INNER JOIN:** Επιστρέφει όλες τις εγγραφές στις οποίες υπάρχει κοινό γνώρισμα <u>και από τους δύο πίνακες!</u>



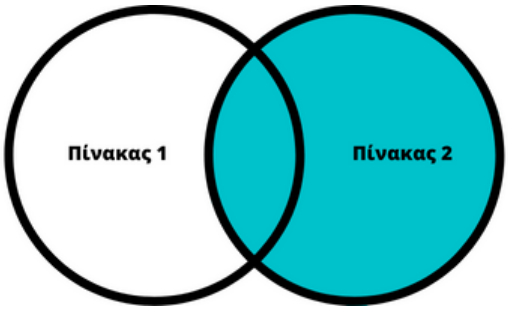
Εικόνα 2 Ένωση πινάκων Inner Join

- **OUTER JOINS:**
  - **LEFT JOIN:** Επιστρέφει όλες τις γραμμές του Πίνακα 1 στα αριστερά και τις σχετικές εγγραφές του Πίνακα 2 στα δεξιά που ικανοποιούν την συνθήκη.



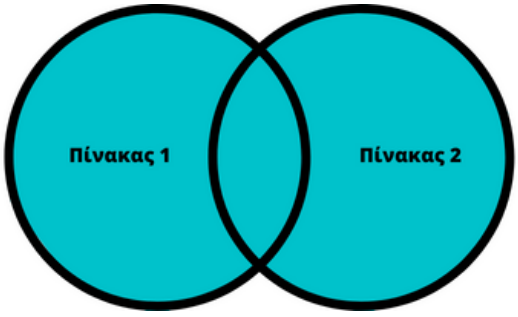
Εικόνα 3 Ένωση πινάκων Left Outer Join

- **RIGHT JOIN:** Επιστρέφει όλες τις γραμμές του Πίνακα 2 στα δεξιά και τις σχετικές εγγραφές του Πίνακα 1 στα αριστερά που πληρούν τις προϋποθέσεις που έχουμε ορίσει.



Εικόνα 4 Ένωση πινάκων Right Outer Join

- **FULL JOIN:** Επιστρέφει όλες τις εγγραφές των πινάκων όταν υπάρχει το κοινό γνώρισμα σε έναν τουλάχιστον πίνακα από τους δύο που συνενώνονται.



Εικόνα 5 Ένωση πινάκων Full Join

Ανάλογα με τα ερωτήματα που έχουμε θέσει στην βάση δεδομένων αλλά και τα αποτελέσματα που θέλουμε να προβάλλουμε μέσω διαφορετικών πινάκων θα πρέπει να χρησιμοποιήσουμε και τον αντίστοιχο τύπο συνένωσης από αυτούς που αναφέρθηκαν παραπάνω.

Επιπρόσθετα αξίζει να αναφέρουμε και τον τύπο ένωσης **CROSS JOIN** ο οποίος δημιουργεί ένα καρτεσιανό γινόμενο το οποίο όταν εκτελείται προβάλει όλους τους δυνατούς συνδυασμούς μεταξύ των εγγραφών των δύο πινάκων.

Παρότι οι συνενώσεις είναι ένα πολύ χρήσιμο εργαλείο για την προβολή των αποτελεσμάτων απαιτείται ιδιαίτερη προσοχή στην χρήση τους ώστε να μην αναπαράγονται διπλότυπες εγγραφές ή να μην εξαιρούνται γραμμές που πληρούν τις προϋποθέσεις του ερωτήματος και περιέχουν το κοινό γνώρισμα.

2.3.Σύνδεση της βάσης δεδομένων με την ιστοσελίδα

Η σύνδεση της βάσης δεδομένων αποτελεί μια απαραίτητη διαδικασία όταν θέλουμε να διαχειριζόμαστε και να έχουμε άμεση πρόσβαση στην βάση μέσα από κάποια εφαρμογή ή ιστοσελίδα με την χρήση δηλαδή ενός εξωτερικού παράγοντα.

Η σύνδεση αυτή επιτυγχάνεται συνήθως χρησιμοποιώντας κάποια υπάρχουσα γλώσσα προγραμματισμού που έχει σχεδιαστεί για να επικοινωνεί με τον server. Μια τέτοια γλώσσα την οποία μάλιστα θα χρησιμοποιήσουμε και στην δική μας πλατφόρμα είναι η PHP.

Ο παρακάτω κώδικας είναι αυτός που συντάξαμε για την σύνδεση της βάσης δεδομένων που κατασκευάσαμε για την πλατφόρμα μας medical:

```php
<?php

/**
 * ΣΥΝΔΕΣΗ ΒΑΣΗΣ ΔΕΔΟΜΕΝΩΝ ΜΕ ΤΗΝ ΠΛΑΤΦΟΡΜΑ
 * Αυτή είναι η κλάση που περιέχει τον constructor για το αντικείμενο ραντεβού
 */

class Appointment {
    public $base_url = 'http://localhost/medical/';
    public $connect;
    public $query;
    public $statement;
    public $now;
```

```
    public function __construct(){
        $this->connect = new PDO("mysql:host=localhost;dbname=medical", "root",
"");

        date_default_timezone_set('Europe/Athens');

        session_start();

        $this->now = date("Y-m-d H:i:s",  STRTOTIME(date('h:i:sa')));
    }
    function execute($data = null){
        $this->statement = $this->connect->prepare($this->query);
        if($data)
        {
            $this->statement->execute($data);
        }
        else
        {
            $this->statement->execute();
        }
    }

    function row_count(){
        return $this->statement->rowCount();
    }

    function statement_result(){
        return $this->statement->fetchAll();
    }

    function get_result(){
        return $this->connect->query($this->query, PDO::FETCH_ASSOC);
    }
}

?>
```

Παρότι στα παρακάτω κεφάλαια θα αναλύσουμε διεξοδικά τον τρόπο με τον οποίο συντάσσονται τα αρχεία PHP εντούτοις θα κάνουμε μια πρώτη προσπάθεια να κατανοήσουμε τον κώδικα που παρουσιάσαμε για την σύνδεση της βάσης.

Παρατηρούμε λοιπόν ότι σε ένα αρχείο PHP το οποίο ξεκινά με το αντίστοιχο tag <?php…?> το πρώτο στοιχείο που κατασκευάζουμε είναι ένα αντικείμενο ή αλλιώς μια κλάση. Στην περίπτωση μας αυτό το αντικείμενο είναι η κλάση appointment μέσα στην οποία περιλαμβάνεται ο

κατασκευαστής της κλάσης και ορισμένες βασικές μέθοδοι που θα χρησιμοποιηθούν στα υπόλοιπα αρχεία php.

Το πρώτο στοιχείο της κλάσης είναι ο ορισμός συγκεκριμένων μεταβλητών που απαιτούνται από τις μεθόδους. Στην μεταβλητή $base_url ορίζεται το μοναδικό μονοπάτι δηλαδή η ρίζα του φάκελου μέσα στον οποίο θα υπάρχουν τα αρχεία του project μας.

Επίσης καθορίζονται οι παρακάτω μεταβλητές:

- $connect: χρησιμοποιείται για την καταχώρηση των στοιχείων της σύνδεσης με την βάση δεδομένων
- $query: απαιτείται για τον ορισμό συγκεκριμένων ερωτημάτων προς την βάση δεδομένων όπως για παράδειγμα να εμφανίσουμε μια λίστα με όλους τους ασθενείς που έχουν κλείσει ραντεβού
- $statement:ορίζεται ως η δήλωση για την εκτέλεση συγκεκριμένων λειτουργιών όπως η κλήση της μεθόδου fetch_all() η οποία θα ανακαλέσει όλες τις εγγραφές κάποιου πίνακα
- $now: μεταβλητή στην οποία αποθηκεύεται η εκάστοτε ημερομηνία και ώρα σύνδεσης με την βάση.

Έχοντας ορίσει τις μεταβλητές που θα χρησιμοποιήσουμε στο σύνολο των αρχείων PHP του συγκεκριμένου Project, προχωρούμε στον ορισμό της σύνδεσης.

Η σύνδεση πραγματοποιείται μέσα στον κατασκευαστή της κλάσης δηλαδή τον constructor. Με μόλις μια εντολή καταχωρούμε στην μεταβλητή connect ένα αντικείμενο PDO της βάσης δεδομένων με όνομα (dbname) medical και χώρο φιλοξενίας την ρίζα – root του localhost, με λίγα λόγια καταχωρούμε στην μεταβλητή τα στοιχεία της βάσης δεδομένων με την οποία θα συνδεθεί.

```
$this->connect = new PDO("mysql:host=localhost;dbname=medical", "root", "");
```

Η παραπάνω εντολή συντάσσεται βάσει των οδηγιών της γλώσσας PHP όπου αναφέρεται ότι η εντολή σύνδεσης καταχωρείται σε μια μεταβλητή και περιλαμβάνει το όνομα της βάσης, τον χώρο φιλοξενίας καθώς και το όνομα διαχειριστή με τον κωδικό πρόσβασης που έχουμε καταχωρίσει κατά την διάρκεια κατασκευής της βάσης:

```
$database = new PDO('mysql:host=localhost;dbname=test', $user, $password);
```

Έπειτα ορίζουμε ως προεπιλεγμένη ζώνη ημερομηνίας και ώρας την Ευρωπαϊκή με πρωτεύουσα την Αθήνα ώστε όταν χρειαστεί να υπολογίσουμε και να καταχωρήσουμε την εκάστοτε ημερομηνία και ώρα στην μεταβλητή now να έχουμε εντοπίσει την σωστή ζώνη ώρας για να μην προκληθούν τυχόν προβλήματα με την εγγραφή στοιχείων στην βάση δεδομένων.

Όπως παρατηρούμε πριν από τον ορισμό της ζώνης ώρας υπάρχει η εντολή session_start(). Αυτή η εντολή αναλαμβάνει να εκτελέσει την εκκίνηση των open και read handlers. Οι handlers είναι στην ουσία τα εργαλεία που μας βοηθούν στην ανάγνωση της εισόδου και εξόδου των δεδομένων. Με την εκκίνηση αυτών των εργαλείων ελέγχονται τα αρχικά ορίσματα των μεταβλητών, ξεκινά η κατασκευή των κλάσεων, η σύνδεση της βάσης δεδομένων και φυσικά ο έλεγχος των στοιχείων που δίνονται στις φόρμες login και signup της πλατφόρμας που αφορούν την σύνδεση και την εγγραφή των χρηστών.

Η εντολή session_start() μας επιστρέφει μια δυαδική – Boolean απάντηση, true ή false. Σε περίπτωση που είναι true τότε η εκκίνηση του συστήματος έγινε με επιτυχία ενώ στην περίπτωση που είναι false τότε υπήρξαν προβλήματα.

Η μέθοδος execute() είναι η βασική μέθοδος εκτέλεσης των ερωτημάτων και των δηλώσεων. Στην παρακάτω εντολή λαμβάνουμε ένα ερώτημα που έχει καταχωρηθεί στην μεταβλητή query. Αυτό το ερώτημα μετατίθεται στην μεταβλητή connect με την οποία θα γίνει η σύνδεση στην βάση δεδομένων και θα αναζητηθεί ο συγκεκριμένος πίνακας του ερωτήματος της query. Όταν καταφέρει να συνδεθεί και να εντοπίσει τον πίνακα που απαιτείται τότε το αποτέλεσμα θα καταχωρηθεί στην μεταβλητή statement. Έπειτα πραγματοποιείται ανάκληση της μεθόδου και η εκτέλεση της διαφέρει ανάλογα με το περιεχόμενο της παραμέτρου data. Αν η παράμετρος data δεν είναι κενή null τότε εκτελείται η δήλωση statement χρησιμοποιώντας τα στοιχεία data, σε διαφορετική περίπτωση εκτελείται χωρίς την παράμετρο.

Η μέθοδος row_count() είναι μια απλή συνάρτηση που εκτελείται για τον υπολογισμό των εγγραφών ενός πίνακα όταν πραγματοποιείται ένα ερώτημα.

Η μέθοδος statement_result() καταχωρεί στην μεταβλητή statement την επιλογή όλων των εγγραφών που υπολογίστηκαν από κάποιο ερώτημα.

Τέλος η μέθοδος get_result() είναι μια συνάρτηση getter, που όταν εκτελείται μας επιστρέφει το αποτέλεσμα μιας δήλωσης και κατά συνέπεια το αποτέλεσμα ενός ερωτήματος μέσω της σύνδεσης με την χρήση της μεταβλητής connect.

2.4. Δημιουργία ερωτημάτων – Queries

Η δημιουργία των ερωτημάτων όπως και η σύνδεση της βάσης δεδομένων γίνεται μέσω της γλώσσας PHP. Στην δική μας περίπτωση για να έχουμε μεγαλύτερη ευελιξία και οργάνωση στα αρχεία του κώδικα μας, αποφασίσαμε τα διάφορα ερωτήματα προς την βάση δεδομένων να συνταχθούν σε ξεχωριστά αρχεία ανάλογα με την λειτουργία που επιτελούν για παράδειγμα η ανανέωση των προσωπικών στοιχείων ενός ασθενή είναι διαφορετικό ερώτημα και άρα διαφορετικό αρχείο από την ανανέωση των στοιχείων ενός γιατρού. Τα παραπάνω αρχεία που αποτελούν τα ερωτήματα έχουν αποθηκευτεί στον υποφάκελο actions του φακέλου classes.

Κάθε αρχείο ερωτήματος φέρνει το ίδιο όνομα με το αντίστοιχο του αρχείο JavaScript που το καλεί με την διαφορά ότι στο τέλος του ονόματος αναγράφεται η λέξη action ώστε να μπορούμε να ξεχωρίσουμε ευκολότερα τα αρχεία ερωτημάτων από τα αρχεία JavaScript που τα καλούν.

Ενδεικτικά θα αναφερθούμε στο αρχείο του ερωτήματος ανανέωσης των προσωπικών στοιχείων του ασθενή, Edit_patient_profile_action.php, με σκοπό την κατανόηση του τρόπου σύνταξης ενός απλού ερωτήματος.

```php
<?php

//action.php

include('../appointment.php');

$object = new Appointment;


if(isset($_POST["action"]))
{

    if($_POST['action'] == 'edit_profile')
    {
        $data = array(
            ':patient_password'         =>  $_POST["patient_password"],
            ':patient_name'             =>  $_POST["patient_name"],
            ':patient_surname'          =>  $_POST["patient_surname"],
            ':patient_birthdate'        =>  $_POST["patient_birthdate"],
            ':patient_gender'           =>  $_POST["patient_gender"],
            ':patient_address'          =>  $_POST["patient_address"],
            ':patient_mobile'           =>  $_POST["patient_mobile"],
            ':patient_phone'            =>  $_POST["patient_phone"],
            ':patient_status'           =>  $_POST["patient_status"]
```

```
        );

        $object->query = "
        UPDATE patient
        SET patient_password = :patient_password,
        patient_name = :patient_name,
        patient_surname = :patient_surname,
        patient_birthdate = :patient_birthdate,
        patient_gender = :patient_gender,
        patient_address = :patient_address,
        patient_mobile = :patient_mobile,
        patient_phone = :patient_phone,
        patient_status = :patient_status
        WHERE patient_id = '".$_SESSION['patient_id']."'
        ";

        $object->execute($data);

        $_SESSION['success_message'] = '<div class="alert alert-success">Η αλλαγη
στοιχείων καταχωρήθηκε!</div>';

        echo 'done';
    }
}
```

Παρατηρούμε ότι το πρώτο πράγμα που δημιουργούμε είναι ένα νέο αντικείμενο της κλάσης appointment και αυτό γιατί όπως αναφέραμε προηγουμένως μέσα στην κλάση appointment είναι όλες οι μέθοδοι που θα χρειαστούμε για να εκτελεστεί το ερώτημα που θέτουμε.

Με την δήλωση if(isset($_POST[“action”])){…} ρωτάμε το σύστημα αν έχει οριστεί η μεταβλητή $_POST με παράμετρο action δηλαδή αν έχει ενεργοποιηθεί η φόρμα με τα στοιχεία του ασθενή σε επίπεδο frontend που λειτουργεί ως είσοδος δεδομένων. Αν έχει ενεργοποιηθεί τότε η επόμενη ερώτηση στο σύστημα είναι αν η ενεργοποίηση αυτή ισούται με την επιλογή της μεταβολής των στοιχείων, δηλαδή αν έχει πατηθεί από τον χρήστη το κουμπί edit_profile.

Στην περίπτωση που ο χρήστης έχει πατήσει το κουμπί και θέλει να μεταβάλλει τα στοιχεία του τότε δηλώνεται ένας πίνακας array με όνομα μεταβλητής data μέσα στον οποίο ορίζονται τα πεδία στα οποία θα τοποθετηθούν τα νέα στοιχεία που εισάγει ο χρήστης.

Έπειτα δημιουργείται το ερώτημα της ανανέωσης των στοιχείων στην βάση δεδομένων μέσω της μεταβλητής query. Η εντολή query παίρνει ως όρισμα την παρακάτω εντολή:

```
UPDATE patient
        SET patient_password = :patient_password,
        patient_name = :patient_name,
        patient_surname = :patient_surname,
        patient_birthdate = :patient_birthdate,
        patient_gender = :patient_gender,
        patient_address = :patient_address,
        patient_mobile = :patient_mobile,
        patient_phone = :patient_phone,
        patient_status = :patient_status
WHERE patient_id = '".$_SESSION['patient_id']."
```

Μέσω της εντολής update δηλώνεται η ανανέωση των πεδίων του πίνακα patient με τα στοιχεία που έχει εισάγει ο χρήστης στην φόρμα με την προϋπόθεση – WHERE το id του ασθενή στον πίνακα να είναι ίδιο με το id του ασθενή που έχει συνδεθεί στην πλατφόρμα.

Έπειτα καλείται η μέθοδος execute με παράμετρο τον πίνακα data ώστε να εκτελεστεί το παραπάνω ερώτημα και να ανανεωθούν τα στοιχεία του χρήστη στον πίνακα βάσει των στοιχείων που έχει δώσει στο array data.

Τέλος τυπώνεται το μήνυμα «Η αλλαγή στοιχείων καταχωρήθηκε!» μέσω της μεταβλητής message στην περίπτωση που το ερώτημα έχει εκτελεστεί σωστά. Με την εντολή echo τυπώνεται ένα διπλό μήνυμα σε backend επίπεδο που επιβεβαιώνει την σωστή καταχώρηση.

Με τον ίδιο τρόπο εκτελούνται όλα τα ερωτήματα που έχουμε θέσει στην πλατφόρμα μας.

3. Η ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ CLIENT-SERVER PHP

Στο κεφάλαιο αυτό θα ασχοληθούμε με την γλώσσα προγραμματισμού PHP, η οποία χρησιμοποιείται κατά κύριο λόγο για την κατασκευή μιας σελίδας αλλά και την δημιουργία της σημαντικής επικοινωνίας μεταξύ της σελίδας και του server στον οποίο θα καταχωρηθεί ο ιστότοπος.

Η PHP είναι μια γλώσσα ευρέως γνωστή, δημοφιλής και κατά γενική ομολογία αρκετά εύκολη για κάποιον προγραμματιστή σε μέτριο επίπεδο γνώσεων και δεξιοτήτων που θέλει να ασχοληθεί με την ανάπτυξη διαδικτυακών εφαρμογών και ιστοσελίδων. Είναι μια αντικειμενοστραφής γλώσσα με βασικότερες λειτουργίες τη σύνδεση και ανταλλαγή πληροφοριών με μια βάση δεδομένων, την

εκτέλεση κώδικα στο διακομιστή και την προβολή των πληροφοριών στο χρήστη χωρίς αυτός να ξέρει τι γίνεται στο παρασκήνιο, καθώς επίσης και την εκτέλεση των εντολών ενός προτύπου PHP.

Στα παρακάτω κεφάλαια θα παρουσιάσουμε τον τρόπο με τον οποίον συντάσσεται ένα αρχείο PHP, σε τι ακριβώς χρησιμεύει και τι μπορούμε να υλοποιήσουμε με την χρήση του.

3.1. Τρόπος σύνταξης των αρχείων PHP

Πριν προβούμε στην παρουσίαση του τρόπου σύνταξης ενός αρχείου PHP θα πρέπει πρώτα να δούμε πως μπορούμε να δημιουργήσουμε ένα τέτοιο αρχείο. Ο τρόπος είναι πολύ εύκολος. Σε έναν οποιοδήποτε editor, (στην παρούσα πτυχιακή για τις ανάγκες δημιουργίας της πλατφόρμας χρησιμοποιήσαμε τον editor Visual Code), επιλέγουμε δημιουργία νέου αρχείου και το αποθηκεύουμε στον υπολογιστή μας με την επέκταση .**php** ή .**php4** ή .**phtml**.

Έτσι απλά έχουμε δημιουργήσει ένα αρχείο PHP το οποίο θα πρέπει να τονίσουμε ότι κατά την σύνταξη του θα δούμε ότι μοιάζει πολύ με τα αρχεία HTML.

Αμέσως μετά την δημιουργία των αρχείων PHP θα πρέπει να προβούμε στην εκκίνηση και την ρύθμιση ενός διακομιστή, δηλαδή ενός «πακέτου» που περιλαμβάνει τους μεταγλωττιστές για τα αρχεία PHP και τον χειριστή – μεσάζοντα μεταξύ της σελίδας και της βάσης δεδομένων που επικοινωνούν μέσω της γλώσσας PHP. Ο πιο γνωστός και ευρέως διαδεδομένος και εύχρηστος διακομιστής είναι ο Apache ο οποίος χρησιμοποιείται σε όλα σχεδόν τα γνωστά λειτουργικά συστήματα όπως Linux / Windows / Mac OS.

Ο συνδυασμός του διακομιστή Apache με το λειτουργικό σύστημα Linux, το σύστημα βάσης δεδομένων MySQL και την PHP ονομάζεται LAMP από τα αρχικά των συστημάτων που χρησιμοποιούνται. Στην παρούσα πτυχιακή χρησιμοποιήσαμε τον συνδυασμό XAMPP που αποτελείται από το λειτουργικό σύστημα Windows, τον διακομιστή Apache, το σύστημα βάσης δεδομένων MySQL που εγκαθίσταται στον διακομιστή και την PHP.

Μετά από όλες αυτές τις εισαγωγικές πληροφορίες μπορούμε πλέον να παρουσιάσουμε τον τρόπο με τον οποίο συντάσσονται τα αρχεία της PHP έχοντας υπόψιν μας ότι για μια πιο ενδελεχή έρευνα της γλώσσας μπορούμε να επισκεφτούμε την πλατφόρμα της PHP στο σύνδεσμο: www.php.net.

Ο κώδικας PHP μπορεί να γραφτεί εμφωλευμένα μέσα σε κώδικα HTML ή το αρχείο που θα κατασκευάσουμε να αποτελείται εξ’ ολοκλήρου από κώδικα PHP.

Το πρώτο βήμα στην συγγραφή κώδικα PHP είναι η εισαγωγή του tag <?php?>. Μέσα σε αυτό το tag εισάγονται όλες οι εντολές που θέλουμε να εκτελεστούν από τον compiler της PHP. Στο παράδειγμα που ακολουθεί χρησιμοποιούμε την πιο γνωστή εντολή εκτύπωσης ενός string στην οθόνη, την εντολή echo:

```
<?php
echo 'if you want to serve PHP code in XHTML or XML documents, use
these tags';
?>
```

Όπως παρατηρείται η εντολή echo εισάγεται ανάμεσα στο tag PHP, το οποίο μετά την εντολή echo κλείνει με αγγλικό ερωτηματικό ακολουθούμενο από ένα εισαγωγικό.

Στην περίπτωση που θέλουμε να προσθέσουμε δυο διαφορετικά αλφαριθμητικά σε μια δήλωση της εντολής echo τότε χρησιμοποιούμε την μέθοδο γραφής concatenate, στην ουσία εισάγουμε μια τελεία ανάμεσα στις δύο συμβολοσειρές όπως βλέπουμε και στο παράδειγμα:

```
<?php
echo "thr"."ee";            //τυπώνει το αλφαριθμητικό "three"
echo "twe" . "lve";         //τυπώνει το αλφαριθμητικό "twelve"
echo 1 . 2;                 //τυπώνει το αλφαριθμητικό "12"
echo 1.2;                   //τυπώνει το αλφαριθμητικό 1.2
echo 1+2;                   //τυπώνει το αλφαριθμητικό 3
?>
```

Θα πρέπει να τονίσουμε ότι εάν το αρχείο περιέχει μόνο κώδικα PHP τότε το tag php δεν χρειάζεται να κλείσει στο τέλος του αρχείου.

Όπως αναφέραμε και πιο πάνω ο κώδικας PHP μπορεί να εναλλάσσεται με κώδικα HTML δημιουργώντας έτσι ένα συνονθύλευμα εντολών όπως το παρακάτω:

```
<p>This is going to be ignored by PHP and displayed by the browser.</p>
<?php echo 'While this is going to be parsed.'; ?>
<p>This will also be ignored by PHP and displayed by the browser.</p>
```

Το αποτέλεσμα του κώδικα αυτού θα είναι η εκτύπωση στην οθόνη τριών strings, τα δυο μέσω των HTML tags p και το ένα με την εντολή echo. Εάν ανοίξουμε το εργαλείου προβολής κώδικα (inspection) του browser θα παρατηρήσουμε ότι ο κώδικας εμφανίζεται ως αρχείο HTML και ότι δεν υπάρχουν πουθενά τα tags και οι εντολές PHP, τυπώνεται δηλαδή το αποτέλεσμα της εντολής echo στην σελίδα αλλά η ίδια η εντολή δεν εμφανίζεται μέσα στον κώδικα. Αυτό συμβαίνει γιατί ο compiler διαβάζει και εκτελεί τις εντολές PHP στο background αλλά στον browser τυπώνει το

αποτέλεσμα των εντολών μετατρέποντας το σε HTML στοιχείο ώστε να μπορεί να εμφανιστεί στον φυλλομετρητή.

Μια ακόμη παρατήρηση είναι η εισαγωγή ενός string στην εντολή echo στην οποία χρησιμοποιήσαμε τα μονά εισαγωγικά. Δεν υπάρχουν όμως μόνο τα μονά εισαγωγικά, για την ακρίβεια έχουμε τέσσερα διαφορετικά είδη εισαγωγής των strings:

- **Single Quoted – Μονά Εισαγωγικά:** Τα strings τυπώνονται στην οθόνη όπως ακριβώς είναι, κάτι που σημαίνει ότι οι ειδικές χαρακτήρες όπως το n που εισάγει νέα γραμμή δεν θα εκτελέσει την λειτουργία του αλλά θα τυπωθεί σαν απλό string. Εάν θέλουμε οι ειδικοί χαρακτήρες όπως τα μονά εισαγωγικά να τυπωθούν θα πρέπει πριν από αυτά να εισάγουμε τον χαρακτήρα **backslash** \, και έτσι θα έχουμε « \' ». Αν θέλουμε να τυπώσουμε και τον ίδιο τον χαρακτήρα **backslash** \ τότε εισάγουμε δύο κάθετες παύλες ή μια δίπλα στην άλλη « \\ ».

- **Double Quoted – Διπλά Εισαγωγικά:** Χρησιμοποιούνται όταν θέλουμε να λειτουργήσουν και να τυπωθούν η πληθώρα των ειδικών χαρακτήρων και μεταβλητών που έχουμε σε ένα αρχείο PHP. Με διπλά εισαγωγικά ο ειδικός χαρακτήρας \n που αναφέραμε πιο πάνω μπορεί να λειτουργήσει και να εισάγει μια νέα γραμμή. Η εκτύπωση των μεταβλητών γίνεται με την βοήθεια των αγκυλών δεξιά και αριστερά του ονόματος της μεταβλητής το οποίο ορίζεται πάντα με το σύμβολο του δολαρίου:

```
echo "The {$type}s are"
```

- **Συντακτικό Heredoc:** Η σύνταξη αυτή λειτουργεί όπως τα διπλά εισαγωγικά αλλά αντί για τα εισαγωγικά γράφουμε τριπλά ελληνικά εισαγωγικά <<< ακολουθούμενα από έναν identifier END. Δεν απαιτείται να εισάγουμε το backslash \ για την εισαγωγή ειδικών χαρακτήρων. Ο κώδικας σε Heredoc συντάσσεται ως εξής:

```
<?php
echo <<<END
      a
     b
    c
\n
END;
```

Παρατηρείται ότι ο identifier οφείλει να κλείσει μετά το τέλος του string διαφορετικά θα εμφανιστεί μήνυμα σφάλματος στην οθόνη.

- **Συντακτικό Nowdoc:** Σε αντίθεση με το συντακτικό Heredoc, το Nowdoc λειτουργεί όπως τα μονά εισαγωγικά. Εξακολουθεί να χρησιμοποιεί τα τριπλά ελληνικά εισαγωγικά αλλά ο identifier εισάγεται μέσα σε μονά εισαγωγικά. Ο identifier που χρησιμοποιείται είναι συνήθως ο EOD ή EOT. Έτσι γράφουμε:

```
<?php
echo <<<'EOD'
Example of string spanning multiple lines
using nowdoc syntax. Backslashes are always treated literally,
e.g. \\ and \'.
EOD;
```

Το συγκεκριμένο παράδειγμα θα έχει ως αποτέλεσμα να τυπωθεί το παρακάτω στην σελίδα:

Example of string spanning multiple lines

using nowdoc syntax. Backslashes are always treated literally,

e.g. \\ and \'.

Όπως παρατηρούμε οι ειδικοί χαρακτήρες όπως και στο συντακτικό των μονών εισαγωγικών τυπώνονται στην οθόνη ως strings χωρίς να εφαρμόζεται η λειτουργία τους.

Μετά από την αναφορά στις βασικές αρχές σύνταξης ενός αρχείου PHP καθώς και στα διαθέσιμα συντακτικά πρωτόκολλα που μπορούμε να ακολουθήσουμε στο επόμενο κεφάλαιο θα ασχοληθούμε με την πρώτη και κύρια λειτουργία που μπορούμε να επιτελέσουμε, τον ορισμό μεταβλητών.

3.1.1. Ορισμός Μεταβλητών

Για να αποθηκεύσουμε δεδομένα διαφόρων τύπων όπως αλφαριθμητικό, αριθμός, μονός χαρακτήρας, πίνακας ή και αντικείμενο θα πρέπει πρώτα να ορίσουμε τον χώρο αποθήκευσης δηλαδή μια μεταβλητή.

Ο ορισμός των μεταβλητών είναι μια πολύ απλή γραμμή κώδικα η οποία αποτελείται από τον τύπο της μεταβλητής δηλαδή αν είναι ακέραιος, δεκαδικός, πίνακας ή αντικείμενο, ακολουθούμενος από ένα όνομα που θέλουμε να δώσουμε στην μεταβλητή και που δίπλα του εισάγουμε το σύμβολο του δολάριού δηλώνοντας έτσι ότι αυτή είναι μια μεταβλητή με το συγκεκριμένο όνομα.

3.1.2. Δημιουργία μεθόδων και συναρτήσεων

Θα πρέπει να προσθέσουμε ότι ως αντικειμενοστραφής γλώσσα η PHP περιλαμβάνει ως κύριο χαρακτηριστικό την ύπαρξη των συναρτήσεων (functions) για την δημιουργία και την εκτέλεση των απαιτούμενων λειτουργιών της σελίδας. Η ίδια η γλώσσα περιλαμβάνει μια αρκετά μεγάλη λίστα από συναρτήσεις αν και μας δίνεται η δυνατότητα να ορίσουμε και τις δικές μας όπου αυτό απαιτείται.

3.1.3. Δημιουργία κλάσεων αντικειμένων

Όπως αναφέραμε και παραπάνω η γλώσσα προγραμματισμού PHP είναι μια αντικειμενοστραφής γλώσσα κάτι το οποίο σημαίνει ότι μπορούμε να κατασκευάσουμε οποιοδήποτε αντικείμενο απαιτείτε από τα ζητούμενα ενός προγράμματος.

Η δημιουργία των αντικειμένων γίνεται με τον ορισμό των κλάσεων και των κατασκευαστών τους (constructors). Μια κλάση είναι ένα σχέδιο ή αλλιώς blueprint ενός αντικειμένου και ένα αντικείμενο είναι η εφαρμογή ή το instance μιας κλάσης. Για παράδειγμα έστω ότι θέλουμε να κατασκευάσουμε ένα αντικείμενο που αποτελεί ένα φρούτο. Πρώτα θα πρέπει να ορίσουμε την κλάση fruit μέσα στην οποία θα εισάγουμε την συνάρτηση του κατασκευαστή του αντικειμένου (constructor) αλλά και τις συναρτήσεις των μεθόδων μέσα στις οποίες ορίζουμε ποιες δουλείες μπορεί να πραγματοποιήσει ένα αντικείμενο. Επίσης η κλάση περιλαμβάνει και τις μεταβλητές στις οποίες καταχωρούνται τα χαρακτηριστικά του αντικειμένου. Στο δικό μας παράδειγμα μερικά χαρακτηριστικά της κλάσης fruit μπορεί να είναι το χρώμα (color), το μέγεθος (size) ή ακόμα και το βάρος (weight).

Μια κλάση δηλώνεται ως εξής:

```php
<?php
class Fruit {

  //Μεταβλητές
  public $name;
  public $color;


  //Κατασκευαστής ή αλλιώς Constructor
  function __construct($name, $color) {
    $this->name = $name;
    $this->color = $color;
  }
```

```
  //Μέθοδοι
  function get_name() {
    return $this->name;
  }
  function get_color() {
    return $this->color;
  }
}
```

Στον παραπάνω κώδικα παρατηρούμε ότι έχουμε ορίσει τις μεταβλητές name και color που θα χρησιμοποιηθούν για την καταχώρηση του ονόματος και του χρώματος ενός φρούτου, τις μεθόδους get_name και get_color οι οποίες θα χρησιμοποιηθούν για να επιστρέψουν στον χρήστη τις τιμές των μεταβλητών και τέλος τον constructor του αντικειμένου.

Ο constructor αποτελεί την συνάρτηση με την οποία αρχικοποιούμε τις μεταβλητές της κλάσης κατά την δημιουργία του αντικειμένου.

Σε πλήρη αντίθεση με τον constructor υπάρχει και ο destructor όπου στην ουσία είναι μια συνάρτηση η οποία καλείται αυτόματα από την PHP στην περίπτωση που το αντικείμενο καταστραφεί ή το script διακοπεί ή διαγραφεί. Στο παράδειγμα μας θα εισάγουμε τον συγκεκριμένο destructor ο οποίος όταν κληθεί θα τυπώσει στην οθόνη μήνυμα με το όνομα του αντικειμένου:

```
function __destruct() {
    echo "The fruit is {$this->name}.";
  }
```

Επίσης θα παρατηρήσουμε ότι χρησιμοποιείται ιδιαίτερα το διακριτικό this το οποίο θα πρέπει να αναφέρουμε ότι ονομάζεται keyword από την PHP και χρησιμοποιείται για να αναφερόμαστε στο συγκεκριμένο αντικείμενο που κατασκευάζεται από την παρούσα κλάση. Το keyword αυτό μπορεί να χρησιμοποιηθεί μόνο μέσα στις μεθόδους μιας κλάσης.

Θα πρέπει να αναφέρουμε ότι πολλές φορές θα χρειαστεί να ορίσουμε τα δικαιώματα χρήσης των μεταβλητών ή μεθόδων ώστε να μην υπάρξει σύγχυση κατά την κλήση τους, δηλαδή θα πρέπει να προσδιορίσουμε ποιες μεταβλητές ή μέθοδοι μπορούν να κληθούν από μια τρίτη μέθοδο, συνάρτηση ή κλάση. Ο προσδιορισμός αυτός γίνεται με την χρήση τριών διακριτικών μπροστά από το όνομα της μεταβλητής ή της μεθόδου στην εντολή ορισμού της. Τα τρία αυτά διακριτικά ή αλλιώς modifiers είναι τα private, public και protected. Το modifier private ορίζει ότι η συγκεκριμένη μεταβλητή ή μέθοδος μπορεί να κληθεί μόνο από στοιχεία της κλάσης μέσα στην

οποία ανήκει. Το modifier public χρησιμοποιείται ως προεπιλογή και στην ουσία ορίζει ότι η μεταβλητή ή η μέθοδος μπορούν να κληθούν από όλα τα στοιχεία του κώδικα ανεξάρτητα από την κλάση στην οποία ανήκουν. Τέλος το modifier protected βοηθά στην δήλωση μεταβλητών και μεθόδων που μπορούν να χρησιμοποιηθούν μόνο από στοιχεία της κλάσης στην οποία ανήκουν ή στοιχεία υποκλάσεων της αρχικής κλάσης στην οποία έχουν δηλωθεί.

Εδώ παρατηρούμε την σύνταξη των μεταβλητών με την χρήση των modifiers:

```
class Fruit {
  public $name;
  protected $color;
  private $weight;
}
```

Ειδική αναφορά πρέπει να γίνει στην κλάση ή μέθοδο abstract. Με τον όρο abstract κλάση ή μέθοδος εννοούμε όλες εκείνες τις κλάσεις ή μεθόδους που η μητρική τους κλάση περιλαμβάνει μια μέθοδο η οποία για να εκτελέσει τις ενέργειες που της έχουν ανατεθεί χρειάζεται να εκτελεστούν μέθοδοι των υποκλάσεων.

Ο κανόνας που διέπει τον όρο του abstract είναι ο εξής: Κάθε αφηρημένη κλάση πρέπει οπωσδήποτε να περιλαμβάνει μια αφηρημένη μέθοδο ενώ κάθε αφηρημένη μέθοδος αποτελεί μια μέθοδο που ορίζεται αλλά δεν εφαρμόζεται μέσα στον κώδικα.

Οι αφηρημένες κλάσεις και μέθοδοι δηλώνονται με την χρήση του keyword abstract όπως φαίνεται παρακάτω:

```
<?php
abstract class ParentClass {
  abstract public function someMethod1();
  abstract public function someMethod2($name, $color);
  abstract public function someMethod3() : string;
}
?>
```

3.1.4. Κλήση των μεθόδων του αντικειμένου

Έπειτα από τον ορισμό των μεθόδων μιας κλάσης επίκειται η κλήση τους για την εκτέλεση των λειτουργιών του αντικειμένου. Η κλήση των μεθόδων είναι μια εύκολη διαδικασία, αφού αρκεί να αναφέρουμε το όνομα της κλάσης μέσα στην οποία βρίσκεται η μέθοδος ακολουθούμενο από το όνομα της ίδιας της μεθόδου που θέλουμε να καλέσουμε:

```
ClassName::staticMethod();
```

Το όνομα της κλάσης και το όνομα της μεθόδου χωρίζονται μεταξύ τους με διπλή άνω κάτω τελεία. Το όνομα της μεθόδου συνοδεύεται από παρενθέσεις ενώ δεν πρέπει να ξεχάσουμε στο τέλος της κλήσης το σύμβολο του ελληνικού ερωτηματικού υποδηλώνοντας το τέλος της εντολής.

Στο παρακάτω παράδειγμα καλείται η μέθοδος welcome() της κλάσης greeting:

```
<?php
class greeting {
  public static function welcome() {
    echo "Hello World!";
  }
}

// Κλήση Μεθόδου
greeting::welcome();
?>
```

3.1.5. Σύνδεση μεταξύ των αρχείων PHP

Σε αυτήν την παράγραφο θα παρουσιάσουμε την μέθοδο σύνδεσης δύο διαφορετικών αρχείων php που επιτυγχάνεται με την βοήθεια των εντολών include ή require.

Αυτό που πρέπει να γνωρίζουμε είναι ότι οι εντολές include και require αντιγράφουν κάθε κείμενο / κώδικα και markup από ένα αρχείο PHP και τα εισάγουν μέσα στο αρχείο στο οποίο τα καλούμε. Η διαδικασία αυτή είναι πολύ χρήσιμη σε περιπτώσεις επαναλαμβανόμενης χρήσης συγκεκριμένου κώδικα σε πολλά διαφορετικά αρχεία καθώς δεν χρειάζεται να γράψουμε τον κώδικα πολλές φορές αφού παρέχεται πλέον η δυνατότητα επαναχρησιμοποίησης του.

Ιδιαίτερη προσοχή πρέπει να δοθεί στο γεγονός ότι με την χρήση των παραπάνω εντολών ο κώδικας αντιγράφεται στο αρχείο που καλείται πριν ξεκινήσει η εκτέλεση του.

Οι δύο αυτές εντολές είναι ταυτόσημες εκτός από την περίπτωση αποτυχίας εκτέλεσης τους. Στην περίπτωση σφάλματος κάθε εντολή αντιδρά διαφορετικά, έτσι η εντολή require θα παραγάγει ένα μήνυμα σφάλματος και θα σταματήσει την εκτέλεση του script ενώ η εντολή include θα εμφανίσει ένα μήνυμα προειδοποίησης και θα επιτρέψει στο script να συνεχίσει την εκτέλεση του.

Εάν δημιουργούμε κώδικα PHP με την βοήθεια ενός Framework CMS ή μια περίπλοκη εφαρμογή είναι προτιμότερο να χρησιμοποιήσουμε την εντολή require καθώς αυτό θα βοηθήσει στην

διατήρηση της ασφάλειας και της ακεραιότητας της εφαρμογής σε περίπτωση που λείπει το αρχείο που καλούμε μέσω της εντολής.

Μια χρήσιμη εφαρμογή της εντολής include είναι η δημιουργία ξεχωριστών τμημάτων κώδικα για εύκολη και γρήγορη μελλοντική επεξεργασία και τροποποίηση του κώδικα, για παράδειγμα όταν κατασκευάζουμε μια ιστοσελίδα μέσω PHP καλό είναι να δημιουργούμε ξεχωριστά αρχεία για το header, footer και main body καθώς έτσι θα μπορούμε ανά πάσα στιγμή να τροποποιήσουμε τον κώδικα του κάθε τμήματος χωρίς να παρεμβαίνουμε στο υπόλοιπο πρόγραμμα. Έτσι παρέχεται ασφάλεια, οργάνωση μεταξύ των αρχείων και φυσικά ταχύτατη εύρεση του τμήματος του κώδικα που απαιτεί αλλαγές. Τα ξεχωριστά αυτά τμήματα της ιστοσελίδας ενώνονται μεταξύ τους με την εντολή include σε ένα τρίτο αρχείο που παίζει τον ρόλο του κεντρικού τμήματος της ιστοσελίδας ή εφαρμογής αντίστοιχα.

Συντάσσουμε τις εντολές με τον εξής τρόπο:

```
include 'filename';
ή
require 'filename';
```

όπου filename είναι το όνομα του αρχείου που θέλουμε να καλέσουμε συνοδευόμενο από την επέκταση .php.

Αξίζει να τονιστεί ότι το αρχείο που καλείται καθώς και το αρχείο που καλεί οφείλουν να είναι σε μορφή PHP. Στην περίπτωση που ένα από τα δύο αρχεία είναι σε μορφή HTML ή HTM και δεν θέλουμε να αλλάξουμε την επέκταση τους μπορούμε να δημιουργήσουμε στον ριζικό φάκελο του project ένα αρχείο htaccess. Το αρχείο αυτό αναλαμβάνει να μεταφέρει το μήνυμα στον server ότι αρχεία που θα κληθούν να εκτελεστούν και είναι της μορφής HTML ή HTM θα πρέπει να τα αποκωδικοποιήσει και να τα εκτελέσει ο server ώς PHP.

Μέσα στο αρχείο htaccess εισάγουμε την παρακάτω γραμμή κώδικα που αναλαμβάνει να εκτελέσει την διαδικασία που μόλις περιγράψαμε:

```
AddType application/x-httpd-php .html .htm
```

4. Η ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ JAVASCRIPT

Σε αυτό το κεφάλαιο θα αναφερθούμε στην γλώσσα προγραμματισμού JavaScript. Η γλώσσα αυτή είναι σχεδιασμένη για να λειτουργεί ως διαμεσολαβητής μεταξύ των γλωσσών που χρησιμοποιούνται σε επίπεδο πελάτη – frontend και των γλωσσών που λειτουργούν σε επίπεδο server – backend.

Είναι μια γλώσσα αντικειμενοστραφής και μεταγλωττίζεται κατευθείαν στον Web Browser χωρίς να χρειάζεται να παρέμβει κάποιος εξωτερικός υπολογιστής. Στην περίπτωση μας η JavaScript χρησιμοποιείται για να συνδέσει τα απλά php αρχεία που λειτουργούν σε επίπεδο html δηλαδή κατασκευάζουν το layout των ιστοσελίδων με τα αρχεία php που χρησιμοποιούνται για την λειτουργικότητα της πλατφόρμας όπως η εκτέλεση των ερωτημάτων.

Η JavaScript περιέχει η ίδια συναρτήσεις σε αντίθεση με τον κώδικα της γλώσσας html που αφορούν κυρίως στην κλήση των κατάλληλων php αρχείων ανάλογα με την αλληλεπίδραση την οποία έχει κάθε χρήστης με την πλατφόρμα. Με λίγα λόγια χρησιμοποιείται για να δώσει μια λειτουργικότητα ή μια συμπεριφορά στις ιστοσελίδες για παράδειγμα καθορίζει τι θα συμβεί αν πατηθεί ένα κουμπί στην σελίδα.

Θα παρατηρήσουμε ότι στο δικό μας project έχουμε συντάξει τον κώδικα της JavaScript σε μορφή JQuery. Η JQuery αποτελεί μια από τις βασικότερες βιβλιοθήκες της JavaScript η οποία περιέχει μια σειρά από απλουστευμένες συναρτήσεις και μεταβλητές με σκοπό την μείωση του κώδικα που απαιτείται να συνταχθεί προσφέροντας ευκολία και σαφήνεια στην σύνταξη καθώς και άμεση μεταβολή των στοιχείων σε μικρότερο χρονικό διάστημα.

Στις παρακάτω υποπαραγράφους θα παρουσιαστεί με λεπτομέρεια ο τρόπος σύνταξης των αρχείων JavaScript καθώς και η σύνδεση τους με τα υπόλοιπα αρχεία του project.

4.1. Τρόπος σύνταξης των αρχείων JavaScript

Το συντακτικό της γλώσσας JavaScript μοιάζει πολύ με αυτό της C και αυτό γιατί δεν έχει δομές εισόδου και εξόδου (input & output) αλλά στηρίζεται εξ' ολοκλήρου στον κώδικα html.

Ένα αρχείο JavaScript αποτελείται από συναρτήσεις οι οποίες καλούνται και εκτελούνται όταν ο χρήστης αλληλοεπιδράσει με κάποιο στοιχείο της ιστοσελίδας.

Για την κλήση ενός στοιχείου όπως για παράδειγμα ένα κουμπί της ιστοσελίδας χρησιμοποιούμε την έτοιμη συνάρτηση της JavaScript:

```
document.getElementById("demo").innerHTML = "Hello JavaScript";
```

Δηλαδή καλούμε το στοιχείο που έχει id την λέξη "demo" και βρίσκεται μέσα στο document για να του καταχωρήσουμε ως περιεχόμενο το κείμενο «Hello Javascript!».

Με αυτόν τον τρόπο καλούμε οποιοδήποτε στοιχείο θέλουμε βάσει του id του.

Παραπάνω είδαμε ότι για να εμφανίσουμε ένα μήνυμα αλληλοεπιδρώντας με το στοιχείο demo χρησιμοποιήσαμε την εντολή innerHTML. Εκτός όμως από αυτήν την εντολή υπάρχουν και άλλες εντολές εμφάνισης μηνυμάτων που επιτελούν η καθεμιά με τον δικό της τρόπο.

Έτσι έχουμε τις παρακάτω εντολές:

- **innerHTML:** γράφει ένα μήνυμα μέσα στο στοιχείο στο οποίο αναφέρεται
- **document.write():** γράφει ένα μήνυμα στην οθόνη χωρίς να βρίσκεται μέσα σε κάποιο tag της html
- **window.alert():** εμφανίζει ένα εξωτερικό παράθυρο του browser και μέσα εκεί εμφανίζεται το μήνυμα
- **console.log():** εμφανίζει το μήνυμα στην κονσόλα του inspection μέσα δηλαδή στο backend παράθυρο του browser

Στην γλώσσα JavaScript η σύνταξη των μεταβλητών είναι μια εύκολη υπόθεση. Αρκεί να οριστεί ο τύπος της μεταβλητής ακολουθούμενος από ένα όνομα για την μεταβλητή. Ο ορισμός κλείνει χρησιμοποιώντας το ελληνικό σύμβολο «**;**»

```
var x;
```

Για να καταχωρήσουμε έναν αριθμό, αλφαριθμητικό ή αντικείμενο σε μια μεταβλητή αρκεί να γράψουμε το όνομα της μεταβλητής η οποία θα ισούται με την τιμή που θέλουμε:

```
x = 5;
```

Τα αλφαριθμητικά μπορούν να συνταχθούν χρησιμοποιώντας είτε διπλά εισαγωγικά είτε μονά:

```
"John Doe"

'John Doe'
```

Οι operators της JavaScript είναι τα κοινά αριθμητικά σύμβολα με τα οποία μπορούμε να κάνουμε υπολογισμούς (+ - * /) ενώ η καταχώριση σε μια μεταβλητή γίνεται χρησιμοποιώντας το σύμβολο =. Τα σχόλια εισάγονται στον κώδικα με την χρήση της διπλής καθέτου // όταν έχουμε μια γραμμή κώδικα. Στην περίπτωση που θέλουμε να περικλείσουμε πολλές γραμμές κώδικα σε σχόλιο τότε εισάγουμε μπροστά από την πρώτη γραμμή τα σύμβολα /* και μετά την τελευταία γραμμή τα ίδια σύμβολα με αντίθετη σειρά */.

Πρέπει να σημειωθεί ότι η γλώσσα JavaScript είναι case-sensitive δηλαδή επηρεάζεται ο κώδικας από την χρήση κεφαλαίων και μικρών αλφαριθμητικών για παράδειγμα η μεταβλητή **o**noma είναι διαφορετική από την μεταβλητή **O**noma.

Ο ορισμός ενός πίνακα γίνεται με την χρήση ενός τύπου μεταβλητής, ένα όνομα για τον πίνακα και εισαγωγή των στοιχείων του πίνακα μέσα σε αγκύλες:

```
const cars = ["Saab", "Volvo", "BMW"];
```

ενώ η σύνταξη ενός αντικειμένου διαφέρει στον ορισμό των πεδίων του αντικειμένου ακολουθούμενα από τα στοιχεία τους μέσα σε αγκύλες:

```
const person = {firstName:"John", lastName:"Doe", age:50,
eyeColor:"blue"};
```

Στην περίπτωση που δεν γνωρίζουμε τον τύπο της μεταβλητής που χρησιμοποιούμε αρκεί να δηλώσουμε την εντολή typeof και το όνομα της μεταβλητής. Η εντολή θα μας επιστρέψει τον τύπο της μεταβλητής στην οθόνη.

```
typeof "John Doe"     // Returns "string"
typeof 3.14           // Returns "number"
```

4.1.1. Δημιουργία συναρτήσεων

Όπως αναφέραμε και προηγουμένως η γλώσσα JavaScript χρησιμοποιείται κυρίως για την ευκολία που έχει στην δημιουργία και την χρήση συναρτήσεων. Με την χρήση των συναρτήσεων επιταχύνουμε τον τρόπο σύνταξης του κώδικα αφού δεν χρειάζεται να γράφουμε τις ίδιες γραμμές κώδικα ξανά και ξανά, αντιθέτως μπορούμε να χρησιμοποιούμε μια συνάρτηση η οποία αναλαμβάνει να εκτελέσει μια λειτουργία όσες φορές την καλέσουμε.

Ο τρόπος σύνταξης μιας συνάρτησης υλοποιείται με την γραφή της εντολής function και περικλείοντας τον κώδικα που θέλουμε να εκτελεί η συνάρτηση μέσα σε αγκύλες.

```
function όνομα συνάρτησης(παράμετρος1, παράμετρος2, παράμετρος3) {
  // κώδικας που θα εκτελεστεί
}
```

Οι παράμετροι χρησιμοποιούνται όταν για την εκτέλεση του κώδικα της συνάρτησης απαιτούνται εξωτερικά στοιχεία για παράδειγμα αν μια συνάρτηση εμφανίζει τα στοιχεία ενός πίνακα μέσα σε ένα div τότε θα χρειαστεί να περάσουμε σαν παράμετρο στην συνάρτηση τον ίδιο τον πίνακα για να μπορεί ο κώδικας να διαβάσει το περιεχόμενου του και να το εμφανίσει μέσα στο div.

Για να καλέσουμε μια συνάρτηση ώστε να εκτελεστεί θα πρέπει να γράψουμε το όνομα της ακολουθούμενο από τις παρενθέσεις μέσα στις οποίες θα περάσουμε κάποιο στοιχείο σαν παράμετρο. Αν στην συνάρτηση δεν έχουν δηλωθεί παράμετροι τότε κατά την κλήση της δεν εισάγουμε τίποτα μέσα στις αγκύλες:

```
όνομα συνάρτησης(παράμετρος1, παράμετρος2, παράμετρος3);

ή

όνομα συνάρτησης();
```

4.1.2. Δημιουργία αντικειμένων & Ορισμός μεθόδων

Η δημιουργία ενός αντικειμένου δεν είναι τίποτα περισσότερο από την δημιουργία ενός πίνακα με την διαφορά ότι το αντικείμενο περιλαμβάνει συγκεκριμένα πεδία διαφόρων τύπων μεταβλητής καθώς και τις δικές του εσωτερικές συναρτήσεις ή αλλιώς μέθοδοι όπως συχνά αποκαλούνται.

```
const person = {
  firstName: "John",
  lastName: "Doe",
  age: 50,
  eyeColor: "blue"
};
```

Ένα αντικείμενο είναι μια οντότητα σε ένα πρόβλημα που καλούμαστε να λύσουμε. Για παράδειγμα στην δική μας πλατφόρμα μια οντότητα είναι ο ασθενής επομένως θα μπορούσαμε να έχουμε ένα αντικείμενο ασθενής που θα περιλάμβανε ως πεδία τα στοιχεία του όπως το όνομα, το επώνυμο, την ηλικία του κ.α. καθώς και τις μεθόδους του δηλαδή στην ουσία τις λειτουργίες που μπορεί να εκτελέσει ως οντότητα για παράδειγμα να κλείσει ένα ραντεβού, να μεταβάλει τα προσωπικά του στοιχεία, να περιηγηθεί στην λίστα με τους γιατρούς.

Οι μέθοδοι στην ουσία είναι συναρτήσεις που δηλώνονται ως ιδιότητες του αντικειμένου.

Έτσι στο παραπάνω παράδειγμα με το αντικείμενο άνθρωπος μπορούμε να έχουμε ως μέθοδο την εμφάνιση του ονοματεπώνυμού του αντικειμένου:

```
const person = {
  firstName: "John",
  lastName : "Doe",
  id       : 5566,
  fullName : function() {
    return this.firstName + " " + this.lastName;
  }
};
```

Η μέθοδος fullname() επιστρέφει σαν αλφαριθμητικό το όνομα και το επώνυμο του αντικειμένου με κενό ανάμεσα τους. Το keyword this χρησιμοποιείται εσωτερικά μέσα στην δήλωση των μεθόδων αναφερόμενο στο αντικείμενο μέσα στο οποίο βρίσκονται οι μεταβλητές που θέλουμε να χρησιμοποιήσουμε.

Όταν θέλουμε να καλέσουμε μια μέθοδο ενός αντικειμένου σε οποιοδήποτε σημείο του κώδικα αρκεί να γράψουμε το όνομα του αντικειμένου συνοδευόμενο από τελεία και το όνομα της μεθόδου με παρενθέσεις.

4.2.Σύνδεση των αρχείων JavaScript με το αρχείο δημιουργίας ιστοσελίδας HTML

Ένα αρχείο JavaScript μπορεί να συνδεθεί με τρεις διαφορετικούς τρόπους με τα υπόλοιπα αρχεία html από τα οποία καλείται για εκτέλεση.

Τρόπος 1: Συγγραφή κώδικα JavaScript πάνω στα στοιχεία HTML

Ο τρόπος αυτός είναι ο πιο απλουστευμένος και χρησιμοποιείται συνήθως μόνο για παραδείγματα εκπαιδευτικής χρήσης γιατί αν χρησιμοποιηθεί σε κώδικα μεγάλης έκτασης τότε θα δυσκολέψει σε μεγάλο βαθμό την συγγραφή, την μετατροπή και την διόρθωση του κώδικα.

Η λειτουργία JavaScript προστίθεται μέσα στο tag του κάθε στοιχείου html που θέλουμε να μεταβάλλουμε με τον παρακάτω τρόπο:

```
<!DOCTYPE html>
<html>
<body>
<h2>My First JavaScript</h2>
```

```
<button type="button"
onclick="document.getElementById('demo').innerHTML = Date()">
Click me to display Date and Time.</button>

<p id="demo"></p>
</body>
</html>
```

Στο παράδειγμα αυτό δηλώνεται στο κουμπί button ότι αν ο χρήστης το πατήσει τότε το στοιχείο της παραγράφου <p> με id=”demo” θα πάρει ως περιεχόμενο και θα εμφανίσει στην οθόνη την ημερομηνία και την ώρα.

Τρόπος 2: Συγγραφή κώδικα JavaScript μέσα στο αρχείο HTML

Ο δεύτερος τρόπος προτιμάται όταν τα αρχεία είναι σε πολύ απλή μορφή και οι λειτουργίες που επιτελούνται είναι οι βασικές με αποτέλεσμα ο κώδικας να είναι μικρής έκτασης. Ο κώδικας JavaScript μπορεί να προστεθεί μέσα στο tag <head> ή μέσα στο tag <body> ακριβώς πριν κλείσει και σηματοδοτείται το τέλος και η αρχή του με το tag <script>…</script>.

```
<!DOCTYPE html>
<html>
<head>
<link rel=”stylesheet” src=”css/styles.css”>
</head>
<body>
<h2>JavaScript in Body</h2>
<p id="demo"></p>

<script>
document.getElementById("demo").innerHTML = "My First JavaScript";
</script>

</body>
</html>
```

Στο παραπάνω παράδειγμα ο κώδικας JavaScript έχει προστεθεί με το tag του <script> μέσα στο κυρίως σώμα του αρχείου το body.

Ενώ παρακάτω διακρίνεται η προσθήκη του μέσα στο tag <head>:

```
<!DOCTYPE html>
<html>
<head>
<link rel=”stylesheet” src=”css/styles.css”>
```

```
<script>
document.getElementById("demo").innerHTML = "My First JavaScript";
</script>

</head>
<body>
<h2>JavaScript in Body</h2>
<p id="demo"></p>
</body>
</html>
```

Τρόπος 3: Εξωτερικό αρχείο JavaScript με σύνδεση στο αρχείο HTML

Με τον τρίτο τρόπο δημιουργούμε ένα εξωτερικό αρχείο JavaScript κάτι το οποίο συνίσταται στην πλειοψηφία των περιπτώσεων γιατί προσφέρει ευελιξία στην συγγραφή των αρχείων και σωστή οργάνωση. Η κλήση του αρχείου JavaScript γίνεται είτε μέσα στο tag <head> ή μέσα στο κυρίως σώμα <body> ακριβώς πριν κλείσει το tag.

Καλούμε το αρχείο χρησιμοποιώντας το tag <script> και ορίζοντας τον τύπο του αρχείου μαζί με το όνομα και την θέση του μέσα στον φάκελο.

Πρέπει να αναφερθεί ότι τα αρχεία JavaScript δημιουργούνται με την επέκταση .js.

```
<!DOCTYPE html>
<html>
<head>
<link rel=”stylesheet” src=”css/styles.css”>
<script language="JavaScript" type="text/javascript" SRC="name.js">


</head>
<body>
<h2>JavaScript in Body</h2>
<p id="demo"></p>
</body>
</html>
```

```
<!DOCTYPE html>
<html>
<head>
<link rel=”stylesheet” src=”css/styles.css”>

</head>
<body>
```

```
<h2>JavaScript in Body</h2>
<p id="demo"></p>

<script language="JavaScript" type="text/javascript" SRC="name.js">


</body>
</html>
```

Στην περίπτωση που χρησιμοποιήσουμε την βιβλιοθήκη JQuery μέσα στα εξωτερικά αρχεία JavaScript τότε θα πρέπει να καλέσουμε και το συγκεκριμένο αρχείο της βιβλιοθήκης μέσα στο αρχείο html με προσοχή γιατί πάντα θα πρέπει να καλείται πρώτο από το εξωτερικό αρχείο JavaScript, διαφορετικά δεν θα μπορούν εκτελεστούν οι εντολές της βιβλιοθήκης που θα έχουμε χρησιμοποιήσει αφού δεν θα μπορεί να τις βρει καθώς το αρχείο της βιβλιοθήκης θα εκτελείτε μεταγενέστερα.

Επομένως καλώντας και το αρχείο της JQuery η σύνταξη του αρχείου html θα είναι ως εξής:

```
<!DOCTYPE html>
<html>
<head>
<link rel=”stylesheet” src=”css/styles.css”>

</head>
<body>
<h2>JavaScript in Body</h2>
<p id="demo"></p>

<script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.7.2/jquery.min.js">
</script>

<script language="JavaScript" type="text/javascript" SRC="name.js">


</body>
</html>
```

Τέλος πρέπει να τονιστεί ότι το εξωτερικό αρχείο JavaScript συντάσσετε με την μορφή συναρτήσεων, δηλαδή όλες οι εντολές που προσθέσαμε στα παραπάνω παραδείγματα και βρίσκονταν μέσα στο tag <script> τώρα θα τις εισάγουμε μέσα σε μια συνάρτηση function όπως φαίνεται παρακάτω:

```
//Εξωτερικό αρχείο JavaScript test.js
```

```
function myFunction() {
  document.getElementById("demo").innerHTML = "Paragraph changed.";
}
```

```
<!--Αρχείο HTML-->

<body>
<h2>JavaScript in Body</h2>
<p id="demo"></p>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.7.2/jquery.min.js">
</script>
<script language="JavaScript" type="text/javascript" SRC="test.js">
</body>
```

4.3.Πλεονεκτήματα και Μειονεκτήματα της JavaScript

Τα πλεονεκτήματά της JavaScript είναι αρκετά ώστε να την επιλέξει κάποιος προγραμματιστής για την δημιουργία κάποιας εφαρμογής. Μερικά από αυτά είναι:

- Η υποστήριξη της από την πλειοψηφία των browsers και η χρήση της χωρίς επιπλέον plugins, αρκεί μόνο ένας browser από τους δεκάδες που την υποστηρίζουν,
- Η επαλήθευση στον client είναι ταχύτερη με την χρήση της JavaScript αφού δεν χρειάζεται να περιμένουμε για να σταλεί πρώτα η φόρμα στον server με την χρήση κάποιου CGI προγράμματος όπως γινόταν στο παρελθόν, να επαληθευτεί και μετά να εμφανιστούν τυχόν σφάλματα, γίνεται απευθείας από την ίδια την γλώσσα,
- Περισσότεροι συνδεδεμένοι χρήστες στον server καθώς πλέον δεν χρειάζεται να απασχολείται με την επαλήθευση όπως προαναφέραμε,
- Πιο απλή μορφή από την Java και πιο εξελιγμένη από την HTML. Ο συνδυασμός της JavaScript με την HTML δημιουργούν την DHTML τις δυναμικές ιστοσελίδες που υπερέχουν λόγω της λειτουργικότητας τους,
- Μεγάλος όγκος βιβλιογραφίας καθώς είναι μια από τις πιο διαδεδομένες γλώσσες προγραμματισμού για το διαδίκτυο με αποτέλεσμα να μπορεί ένας προγραμματιστής να βρει εύκολα πηγές εκπαιδευτικού υλικού και λύσεις σε προβλήματα που ανακύπτουν.

Από την άλλη μεριά όπως σε κάθε γλώσσα έτσι και στην Javascript υπάρχουν ορισμένα μειονεκτήματα που είναι σημαντικό να αναφερθούν:

- Ύπαρξη πολυπλοκότητας στην κατανόηση του κώδικα καθώς δεν είναι εμφανές απ' την αρχή πως αλληλοεπιδρούν τα αρχεία μεταξύ τους
- Έντονη αργοπορία στο κατέβασμα και στην εκτέλεση των αρχείων όταν αυτά είναι μεγάλης έκτασης
- Παρότι η πλειοψηφία των browsers υποστηρίζει την JavaScript εντούτοις αρκετές φορές υπάρχουν ασυμβατότητες με αποτέλεσμα πολλές φορές να πρέπει να συντάξουμε εναλλακτικούς τρόπους εκτέλεσης των διαφόρων εντολών.

Ως λύσεις ανάγκης ή εναλλακτικούς τρόπους δημιουργίας δυναμικών ιστοσελίδων και εφαρμογών έναντι της JavaScript μπορούμε να προτείνουμε τα frameworks Shockwave, Flash για ανάπτυξη γραφικών και animation, CGI για κατασκευή φόρμας και την γνωστή σε όλους μας Java.

5. Η ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ ΜΟΡΦΟΠΟΙΗΣΗΣ CSS

Για την κατασκευή μιας ιστοσελίδας ή διαδικτυακής εφαρμογής μπορούμε όπως είδαμε να χρησιμοποιήσουμε είτε την γλώσσα προγραμματισμού PHP είτε HTML, αυτό όμως που δεν μπορούμε να αλλάξουμε και επιβάλλεται να χρησιμοποιήσουμε οπωσδήποτε είναι τα αρχεία σε μορφή CSS.

Τα αρχεία μορφοποίησης CSS είναι υπεύθυνα για την γραφή των στοιχείων γραφικού σχεδιασμού μιας ιστοσελίδας, δηλαδή μέσα από αυτά ορίζουμε την εμφάνιση ενός στοιχείου HTML στον φυλλομετρητή. Με τον όρο εμφάνιση εννοούμε για παράδειγμα το χρώμα της γραμματοσειράς, το χρώμα του background, το περίγραμμα ή ακόμα και ο τρόπος κίνησης ενός κουμπιού όταν ο κέρσορας του ποντικιού περνάει από πάνω του. Όλα αυτά τα χαρακτηριστικά ορίζονται μέσω των αρχείων CSS.

Ένας κώδικας της μορφής CSS χαρακτηρίζεται από ευελιξία, προσαρμοστικότητα και εύκολη σύνταξη κάτι το οποίο θα παρουσιάσουμε στην επόμενη παράγραφο.

5.1. Τρόπος σύνταξης των αρχείων CSS

Όπως αναφέραμε τα αρχεία CSS είναι γνωστά για την ευκολία που παρουσιάζουν στην σύνταξη τους. Για να μορφοποιήσουμε ένα οποιοδήποτε στοιχείο HTML αρκεί να γράψουμε τον συντελεστή του στοιχείου ακολουθούμενο από αγκύλες μέσα στις οποίες τοποθετούμε τα ορίσματα.

Τα ορίσματα είναι οι ιδιότητες της εμφάνισης των στοιχείων για παράδειγμα αν θέλουμε να ορίσουμε το μέγεθος της γραμματοσειράς μιας παραγράφου θα χρησιμοποιήσουμε την ιδιότητα ***font-size*** η οποία συνοδεύεται με άνω - κάτω τελεία, τον αριθμό του μεγέθους που θέλουμε με την μονάδα μέτρησης σε px, rem, em, % ή vw και τέλος κλείνει με το σύμβολο του ελληνικού ερωτηματικού.

```
p {
   font-size : 15px;
}
```

Σε αυτήν την περίπτωση κάθε παράγραφος του αρχείου HTML θα έχει μέγεθος γραμματοσειράς 15px. Πολλές φορές μπορεί να χρειαστεί να διαφοροποιήσουμε την μορφοποίηση μεταξύ όμοιων στοιχείων για παράδειγμα μπορεί να θέλουμε μια συγκεκριμένη παράγραφος να έχει μέγεθος γραμματοσειράς 18px αντί για 15. Για να μπορέσουμε να πραγματοποιήσουμε αυτόν τον διαχωρισμό θα πρέπει να χρησιμοποιήσουμε τις κλάσεις ή τα μοναδικά κλειδιά.

Κλάσεις ονομάζουμε τα διακριτικά ή αλλιώς ψευδώνυμα ενός στοιχείου HTML μέσα από τα οποία μας δίνεται η δυνατότητα να καλούμε και να αναφερόμαστε σε ξεχωριστά στοιχεία. Μια κλάση μπορεί να χρησιμοποιηθεί σε παραπάνω από ένα στοιχείο χωρίς περιορισμούς. Έτσι εάν έχουμε τρείς παραγράφους στο αρχείο HTML και εμείς θέλουμε να μορφοποιήσουμε μόνο τις δύο μπορούμε να τους δώσουμε μια κλάση με οποιοδήποτε όνομα θέλουμε. Στο παρακάτω παράδειγμα έχουμε ορίσει ως κλάση την λέξη tomato στις δύο από τις τρείς παραγράφους.

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">

  <title>A Basic HTML5 Template</title>
  <meta name="description" content="A simple HTML5 Template for new
projects.">
  <link rel="icon" href="/favicon.ico">
  <link rel="stylesheet" href="css/styles.css?v=1.0">
</head>
<body>


  <p class=”colorize”> This is the first paragraph </p>
  <p> This is the second paragraph </p>
  <p class=”colorize”> This is the third paragraph </p>
```

```
  <script src="js/scripts.js"></script>
</body>
</html>
```

Στο παράδειγμα μας θέλουμε οι δύο αυτές παράγραφοι που έχουν την κλάση colorize να αποκτήσουν ως χρώμα γραμματοσειράς το tomato. Αυτό θα γίνει καλώντας την κλάση μέσα στο αρχείο CSS στην θέση του συντελεστή p. Έτσι θα έχουμε:

```
.colorize {
     color : tomato;
}
p {
     color : blue;
}
```

Ως αποτέλεσμα στον φυλλομετρητή θα εμφανιστούν η πρώτη και η τρίτη παράγραφος με χρώμα γραμματοσειράς tomato ενώ η δεύτερη με χρώμα μπλε. Αν υπήρχε ως μορφοποίηση μόνο το όρισμα p στο αρχείο CSS τότε όλες οι παράγραφοι θα είχαν μπλε χρώμα τώρα όμως που εισάγουμε μορφοποίηση για τις παραγράφους που περιλαμβάνουν την κλάση, το μπλε χρώμα παρακάμπτεται και την θέση του καταλαμβάνει το tomato.

Ιδιαίτερη προσοχή χρειάζεται στο γεγονός ότι για να αναφερθούμε σε μία κλάση μέσα στο αρχείο CSS χρησιμοποιούμε το όνομα της αφού πρώτα έχουμε προσθέσει μια τελεία.

Πολλές φορές μπορεί να θέλουμε να ξεχωρίσουμε ένα στοιχείο HTML από άλλα όμοια μετατρέποντας το σε μοναδικό. Αυτό μπορεί να επιτευχθεί χρησιμοποιώντας τα μοναδικά κλειδιά ή αλλιώς id.

Προσοχή! Ένα id πρέπει να δοθεί σε ένα μοναδικό στοιχείο, δεν επιτρέπεται η επαναχρησιμοποίηση του σε αντίθεση με τις κλάσεις.

Στο παράδειγμα μας θα δώσουμε στην δεύτερη παράγραφο ως id την λέξη bigger με σκοπό να ορίσουμε μεγαλύτερο μέγεθος γραμματοσειράς μόνο σε αυτό το κείμενο.

```
//Αρχείο HTML

<p class=”colorize”> This is the first paragraph </p>
<p id=”bigger” > This is the second paragraph </p>
<p class=”colorize”> This is the third paragraph </p>
```

```
//Αρχείο CSS
```

```
.colorize {
     color : tomato;
}
#bigger {
     font-size : 25px;
}
```

Παρατηρούμε ότι καλούμε το id έχοντας προσθέσει πριν το όνομα του το σύμβολο της δίεσης. Με την βοήθεια αυτού του συμβόλου ξεχωρίζουμε τις κλάσεις, τα id και τα HTML tags μεταξύ τους. Το συγκεκριμένο μέγεθος γραμματοσειράς θα εισαχθεί μόνο στην δεύτερη παράγραφο που φέρει το μοναδικό id. Εάν θέλουμε να χρησιμοποιήσουμε το ίδιο id μέσα στο ίδιο project τότε θα πρέπει να σιγουρευτούμε ότι θα το εισάγουμε σε στοιχείο διαφορετικού αρχείου HTML, δηλαδή θα πρέπει να καταστεί κατανοητό ότι ίδιο id καταχωρείται μόνο σε στοιχεία που ανήκουν σε διαφορετικά αρχεία HTML.

Σε περίπτωση που για κάποιο λόγο ο compiler αρνείται να παρακάμψει την γενική μορφοποίηση παρότι χρησιμοποιούμε για διαφορετική εμφάνιση τις κλάσεις και τα μοναδικά κλειδιά που έχουμε δημιουργήσει, μπορούμε να χρησιμοποιήσουμε το **keyword !important**. Το keyword αυτό τονίζει στον compiler ότι ο ορισμός μιας ιδιότητας προέχει και οφείλει να παρακάμψει οποιοδήποτε άλλο κανόνα μορφοποίησης.

Έτσι στο παράδειγμα μας με την κλάση colorize παρότι έχουμε ορίσει διαφορετική εμφάνιση για τις δυο παραγράφους που χρησιμοποιούν την κλάση μπορεί ο compiler να εμφανίσει και τις τρείς με μπλε χρώμα. Για να τονίσουμε στο πρόγραμμα ότι θα πρέπει να χρησιμοποιηθεί το χρώμα tomato σε κάποιες από αυτές εισάγουμε το keyword δίπλα από το χρώμα όπως φαίνεται παρακάτω:

```
.colorize {
     color : tomato !important;
}
p {
     color : blue;
}
```

Ένας άλλος τρόπος αποκλεισμού συγκεκριμένων στοιχείων κατά τον ορισμό της μορφοποίησης είναι η χρήση του συντελεστή not().

Ας υποθέσουμε ότι θέλουμε να χρησιμοποιήσουμε το κόκκινο χρώμα σε όλες τις παραγράφους εκτός από την παράγραφο με το μοναδικό id bigger τότε θα πρέπει στον κώδικα CSS να εισάγουμε τον συντελεστή not δίπλα από το στοιχείο HTML p ενώ μέσα στις παρενθέσεις θα τοποθετήσουμε το όνομα του id:

```
p : not ( '#bigger' ) {
   color: red;
}
```

5.2.Σύνδεση του αρχείου μορφοποίησης CSS με το αρχείο HTML

Ο κώδικας CSS μπορεί να είναι ενσωματωμένος μέσα στο αρχείο HTML ή PHP ή να αποτελεί ξεχωριστό αρχείο που καλείτε για να εκτελεστεί.

Στην περίπτωση που είναι ενσωματωμένος υπάρχουν δύο τρόποι να προστίθεται. Ο πρώτος είναι να περικλείετε μέσα στο HTML Tag <style> το οποίο εισάγεται μέσα στο head και ο δεύτερος να τοποθετείται μέσα στο ίδιο το στοιχείο στο οποίο αναφέρεται και ορίζει την μορφοποίηση του. Στο παρακάτω παράδειγμα παρατηρούμε τις διαφορετικές μορφές ενσωματωμένου κώδικα CSS:

```
<!doctype html>

<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">

  <title>A Basic HTML5 Template</title>
  <meta name="description" content="A simple HTML5 Template for new
projects.">
  <link rel="icon" href="/favicon.ico">

  <!--Ενσωματωμένος κώδικας CSS μέσω του tag style-->
  <style>
    body {
      background-color: linen;
    }
    p {
      color: maroon;
      margin-left: 40px;
    }

  </style>


</head>
<body>
  <!--Ενσωματωμένος κώδικας CSS στο κάθε στοιχείο ξεχωριστά-->
  <p style="color:red;"> This is the first paragraph </p>
  <p style="margin-left:15px;"> This is the second paragraph </p>

  <script src="js/scripts.js"></script>
</body>
</html>
```

Ο τρόπος αυτός γραφής του κώδικα CSS δεν είναι ο καλύτερος γιατί μειώνει την ευελιξία, την λειτουργικότητα και την προσαρμοστικότητα ενώ παράλληλα αυξάνει το μέγεθος του αρχείου και την πολυπλοκότητα στην τροποποίηση του. Έτσι λοιπόν για να αποφύγουμε αυτά τα μειονεκτήματα είναι προτιμότερο να δημιουργούμε ένα ξεχωριστό αρχείο CSS το οποίο θα καλείται όποτε θέλουμε από τα αρχεία HTML και PHP μέσω του tag link ή των εντολών require / include.

Ένα πλεονέκτημα που κερδίζουμε από την κατασκευή ενός ξεχωριστού αρχείου CSS είναι η επαναχρησιμοποίηση του ίδιου κώδικα από πολλαπλά αρχεία HTML / PHP παρέχοντας μας την δυνατότητα ορισμού και χρήσης κοινής μορφοποίησης.

Για να συνδέσουμε ένα αρχείο CSS με ένα αρχείο HTML αρκεί να χρησιμοποιήσουμε το tag **<link>** με ιδιότητα την href όπου θα πάρει σαν τιμή το μονοπάτι προορισμού του αρχείου CSS. Το tag **<link>** εισάγεται πάντα μέσα στο head του αρχείου HTML.

```
<link rel="stylesheet" href="mystyle.css">
```

Εάν θέλουμε να εισάγουμε ένα αρχείο CSS μέσα σε έναν εξ' ολοκλήρου κώδικα PHP τότε μπορούμε να χρησιμοποιήσουμε την εντολή include όπως φαίνεται παρακάτω:

```
<style>
<?php include 'CSS/main.css';?>
</style>
```

Παρατηρείται ότι η εντολή include εισάγεται μέσα σε ένα tag <style> και αυτό γιατί όταν εκτελεστεί το αρχείο php ο κώδικας css θα πρέπει να συνταχθεί μέσα σε html tag ώστε να μπορέσει ο compiler να τον αναγνωρίσει και να τον εκτελέσει χωρίς σφάλματα.

Ένας άλλος τρόπος να εισάγουμε κώδικα css μέσα σε ένα php αρχείο είναι να συντάξουμε μέσα σε ένα tag την εντολή get_contents() η οποία όταν εκτελεστεί θα εισάγει όλες τις γραμμές κώδικα του αρχείου css μέσα στην μεταβλητή $css και έπειτα θα τυπωθούν μέσα στο tag <style> από την εντολή echo:

```
<style type="text/css">
<?php
$css = file_get_contents('CSS/main.css');
echo $css;
?>
</style>
```

5.3.Σύνταξη αρχείου μορφοποίησης στην απλουστευμένη μορφή SASS

Όπως είδαμε και παραπάνω η σύνταξη ενός αρχείου CSS είναι παράλληλα τόσο εύκολη όσο και χρήσιμη. Η μορφοποίηση μιας ιστοσελίδας ή μιας διαδικτυακής εφαρμογής βασίζεται εξ'ολοκλήρου πάνω σε κώδικα CSS.

Πολλές φορές όμως επειδή ένα αρχείο CSS μπορεί να περιλαμβάνει υπερβολικά πολλά εμφωλευμένα tags και να καθίσταται δύσκολος ο διαχωρισμός των διαφορετικών στοιχείων και στυλ μορφοποίησης, προκειμένου να απλουστεύσουμε τον τρόπο γραφής των αρχείων χρησιμοποιούμε την συμπιεσμένη μορφή σύνταξης SASS.

Τα αρχεία SASS έρχονται ως επέκταση της γλώσσας CSS με σκοπό να λύσει το πρόβλημα του προσδιορισμού και της γραφής εμφωλευμένων tags αφού πλέον μπορούμε να γράψουμε κάθε εμφωλευμένο tag μέσα στο προηγούμενο γονικό tag αντί να τα συντάσσουμε το ένα δίπλα στο άλλο δημιουργώντας έτσι δυσανάγνωστα κομμάτια κώδικα.

Μια επιπλέον βοήθεια που παρέχει η μορφή SASS στην γλώσσα CSS είναι η ύπαρξη υποστήριξης μεταβλητών και μαθηματικών πράξεων.

Η μορφή SASS συνυπάρχει με την SCSS, μια εξίσου απλουστευμένη μορφή που διαφοροποιείται στα εξής σημεία:

SCSS	SASS
Νέο συντακτικό	Παλιότερη μορφή σύνταξης
Επέκταση αρχείου σε **.scss**	Επέκταση αρχείου **.sass**
Σύνταξη παρόμοια με τα αρχεία CSS	Σύνταξη παρόμοια με την γλώσσα Ruby
Δεν απαιτείται η χρήση κενών & νέων γραμμών	Αυστηρή χρήση κενών (tabs) και νέων κενών γραμμών για την σύνταξη εμφωλευμένων tags
Απαιτούνται οι αγκύλες **{}** και το σύμβολο του ελληνικού ερωτηματικού **";"** (semicolons)	Δεν χρησιμοποιούνται αγκύλες **{}** και το σύμβολο ελληνικού ερωτηματικού **";"** (semicolons)

Η χρήση της είναι πιο περιορισμένη αλλά παρέχεται εξίσου πολυπληθές documentation	Παρέχεται μεγαλύτερο documentation ενώ υπάρχει μεγάλη κοινότητα προγραμματιστών για την παροχή τεχνικής υποστήριξης

Πίνακας 3 Διαφορές μεταξύ SASS και SCSS σύνταξης μορφοποίησης.

Στο παρακάτω παράδειγμα βλέπουμε την διαφορά σύνταξης μεταξύ CSS, SCSS & SASS:

CSS:

```
nav ul {
  margin: 0;
  padding: 0;
  list-style: none;
}
nav li {
  display: inline-block;
}
nav a {
  display: block;
  padding: 6px 12px;
  text-decoration: none;
}
```

SCSS:

```
nav {
  ul {
    margin: 0;
    padding: 0;
    list-style: none;
  }

  li { display: inline-block; }

  a {
    display: block;
    padding: 6px 12px;
    text-decoration: none;
  }
}
```

SASS:

```
nav
  ul
```

```
    margin: 0
    padding: 0
    list-style: none

  li
    display: inline-block

  a
    display: block
    padding: 6px 12px
    text-decoration: none
```

5.3.1. Δημιουργία εμφωλευμένων αρχείων μορφοποίησης SCSS – Particles

Αξίζει να σημειωθεί ότι με την βοήθεια της μορφής SCSS μπορούμε να διαχωρίσουμε συγκεκριμένα κομμάτια κώδικα σε διαφορετικά αρχεία με σκοπό τον καλύτερο καταμερισμό, εκτέλεση και σύνταξη των διαφορετικών στυλ μορφοποίησης. Αυτό επιτυγχάνεται με την δημιουργία των αρχείων particles τα οποία ενώνουμε καλώντας τα σε ένα κεντρικό αρχείο SCSS.

Κάθε αρχείο particle ορίζεται με την προσθήκη μιας κάτω παύλας μπροστά από το όνομα του αρχείου το οποίο έχει επέκταση .scss. Το particle καλείται στο κεντρικό αρχείο SCSS χρησιμοποιώντας την εντολή import και ακολουθούμενη με το όνομα του αρχείου μέσα σε εισαγωγικά χωρίς την κάτω παύλα και την επέκταση.

Έχοντας τα παραπάνω κατά νου καταλαβαίνουμε ότι μπορούμε να μορφοποιήσουμε μια ιστοσελίδα ή διαδικτυακή εφαρμογή διαχωρίζοντας κάθε βασικό στοιχείο της σε ξεχωριστό αρχείο particle. Έτσι μπορούμε να έχουμε ένα διαφορετικό αρχείο particle για το header, το footer, την μορφοποίηση των γραμματοσειρών, τον ορισμό των περιθωρίων της σελίδας. Τα διαφορετικά αυτά αρχεία particles αποθηκεύονται σε ένα κοινό φάκελο που περιλαμβάνει την μορφοποίηση της σελίδας και καλούνται σε ένα κεντρικό αρχείο μορφοποίησης scss που βρίσκεται στον ίδιο φάκελο.

Παρακάτω φαίνεται ο καθορισμός των αρχείων και η κλήση τους από το κεντρικό αρχείο SCSS:

```
Styles -> Φάκελος των στυλ μορφοποίησης
├── _header.scss -> Particle αρχείο SCSS για το header
├── _footer.scss -> Particle αρχείο SCSS για το footer
├── _typography.scss -> Particle αρχείο SCSS για τις γραμματοσειρές
├── _margins.scss -> Particle αρχείο SCSS για τα περιθώρια
```

```
└── styles.scss -> Κεντρικό αρχείο που καλεί τα particles
```

Η κλήση των αρχείων particles από το κεντρικό αρχείο μορφοποίησης SCSS είναι ως εξής:

```
import 'header';
import 'footer';
import 'typography';
import 'margins';
```

Όταν θα κληθεί το κεντρικό αρχείο SCSS να εκτελεστεί τότε ο compiler θα αναλάβει να καλέσει τα αρχεία particles μέσω της εντολής import και στην συνέχεια θα τα μεταγλωττίσει σε κώδικα CSS κατασκευάζοντας ένα αρχείο CSS που θα έχει το ίδιο όνομα με το κεντρικό αρχείο SCSS και θα περιλαμβάνει όλο τον κώδικα των particles σε μορφή CSS στην σειρά με την οποία κλήθηκαν μέσω της import. Τέλος θα εκτελέσει το νέο παραγόμενο αρχείο CSS και θα αποτυπώσει την μορφοποίηση στον browser.

Η κατασκευή και η χρήση των particles είναι ο πιο χρήσιμος τρόπος μορφοποίησης καθώς μας δίνει την δυνατότητα να συντάσσουμε και να τροποποιούμε τα διαφορετικά στυλ πιο γρήγορα και με μεγαλύτερη οργάνωση ενώ η εύρεση σφαλμάτων μετατρέπεται σε μια εύκολη διαδικασία αφού η αναζήτηση των λαθών πραγματοποιείται σε μικρότερα τμήματα κώδικα.

6. ΤΟ FRAMEWORK BOOTSTRAP

Έχοντας παρουσιάσει στο παραπάνω κεφάλαιο τον λόγο αλλά και τον τρόπο που χρησιμοποιούμε και συντάσσουμε τα αρχεία CSS, θα προσπαθήσουμε να παρουσιάσουμε σε ένα πλήρη οδηγό το Framework Bootstrap το οποίο έχει δημιουργηθεί για να μας διευκολύνει σε υπέρμετρο βαθμό στην ανάπτυξη των αρχείων CSS και κατά συνέπεια στην δημιουργία ιστοσελίδων με ένα όμορφο και ανταποκρίσιμο γραφικό περιβάλλον με μια πληθώρα διαστάσεων διαφορετικών συσκευών.

Το Framework Bootstrap αναπτύχθηκε για πρώτη φορά από τους προγραμματιστές Mark Otto και Jacob Thornton με σκοπό να χρησιμοποιηθεί στο Twitter. Αξίζει να σημειωθεί ότι πριν το Bootstrap αρκετές βιβλιοθήκες είχαν αναπτυχθεί και χρησιμοποιηθεί για την βέλτιστη ανάπτυξη της διεπαφής μεταξύ χρήστη και συστήματος όμως με την πάροδο του χρόνου παρατηρήθηκε ότι η χρήση πολλαπλών και ανομοιογενών βιβλιοθηκών οδήγησε σε αντιφάσεις και υψηλή φορολογική επιβάρυνση συντήρησης των συστημάτων. Χρειαζόταν άμεσα ένα σύνολο κανόνων, συναρτήσεων και κλάσεων CSS που θα στόχευαν στην ομοιομορφία των διεπαφών και στην βέλτιστη ανταπόκριση τους όταν παρουσιάζονταν σε διαφορετικές συσκευές. Χιλιάδες

προγραμματιστές προχώρησαν στην άμεση χρήση του Bootstrap για την ανάπτυξη ιστοσελίδων και διαδικτυακών εφαρμογών μετά την πρώτη επίσημη παρουσίαση του ρόλου του Framework και των δυνατοτήτων του μετά το πρώτο HackWeek του Twitter. Για την ιστορία το Twitter Bootstrap ξεκίνησε επίσημα την κυκλοφορία του ως λογισμικό ανοιχτού κώδικα τον Αύγουστο του 2011 ενώ μόλις ένα χρόνο αργότερα, τον Φεβρουάριο του 2012 θεωρούνταν ήδη το πιο δημοφιλές έργο ανάπτυξης στο Github.

6.1.Χαρακτηριστικά, Δομή και Λειτουργία του Bootstrap

Σήμερα το Bootstrap θεωρείται το πιο ολοκληρωμένο σύνολο προτύπων για την ανάπτυξη κώδικα CSS.

Σύμφωνα με την διεθνή ορολογία το Bootstrap αποτελεί μια συλλογή εργαλείων ανοιχτού κώδικα δηλαδή ελεύθερο λογισμικό στο οποίο ο οποιασδήποτε προγραμματιστής που έχει τις γνώσεις μπορεί να συνεισφέρει στην ανάπτυξη επιπλέον χρήσιμων εργαλείων που θα χρησιμοποιηθούν για την δημιουργία ιστοσελίδων και διαδικτυακών εφαρμογών. Η συλλογή αυτή των εργαλείων περιέχει τυπογραφίες των γλωσσών HTML & CSS, κουμπιά πλοήγησης και στοιχεία περιβάλλοντος καθώς και προαιρετικές επεκτάσεις της γλώσσας Javascript. Χαρακτηριστικά παραδείγματα εταιρειών που έχουν χρησιμοποιήσει το Bootstrap για την ανάπτυξη των ιστοσελίδων τους είναι η NASA και το MSNBC.

Ένα από τα κυριότερα πλεονεκτήματα που καθιστούν το συγκεκριμένο Framework ως το πιο βέλτιστο είναι η συμβατότητα του σε μεγάλο βαθμό με όλους τους υπάρχοντες φυλλομετρητές. Παρότι το Bootstrap βασίστηκε για την ανάπτυξη του στην έκδοση του CSS3 το οποίο εισήγαγε σαν έννοιες τις στρογγυλεμένες γωνίες των στοιχείων, την σκίαση και τις κλίσεις οι οποίες συχνά δεν προσαρμόζονται εύκολα σε όλους τους browsers, εντούτοις το Bootstrap μέσω των εργαλείων του ήρθε να βελτιώσει αυτήν την κατάσταση και να υπάρχει ή δυνατότητα συμβατότητας από την πλειοψηφία των φυλλομετρητών. Αυτό όμως δεν σημαίνει ότι το Bootstrap απαιτεί την χρήση των παραπάνω στοιχείων για την λειτουργικότητα του αλλά κατά βάθος την επεκτείνει.

Το πιο σημαντικό χαρακτηριστικό του Framework είναι η δυνατότητα δυναμικής διάταξης των στοιχείων μιας ιστοσελίδας / εφαρμογής. Με τον όρο δυναμική διάταξη εννοούμε την μετατροπή των στοιχείων με σκοπό την βέλτιστη αισθητικά παρουσίαση τους σε διαφορετικές διαστάσεις συσκευών, δηλαδή για παράδειγμα πως θα τοποθετηθεί ένας πίνακας ή ένα πλέγμα ώστε να εμφανίζεται καλαίσθητα σε όλες τις διαφορετικές συσκευές είτε αυτές είναι υπολογιστές, tablet ή

κινητά. Η διαδικασία μετατροπής των στοιχείων μέχρι πρότινος πριν την δημιουργία του Framework αποτελούσε μια ιδιαίτερα κουραστική, επίπονη και δύσκολη εργασία για τους απανταχού προγραμματιστές. Το Bootstrap με το σύνολο των κλάσεων και των συναρτήσεων του ήρθε να διώξει αυτό το σημαντικό εμπόδιο από την ανάπτυξη ιστοσελίδων αφού η δυναμική διάταξη παράγεται με την χρήση μονολεκτικών εντολών μειώνοντας τον χρόνο εργασίας σε μερικά δευτερόλεπτα.

Όπως αναφέραμε πιο πάνω τον κώδικα του Framework μπορούμε να βρούμε στο Github που μας παρέχεται η δυνατότητα να συνεισφέρουμε με τις γνώσεις μας στην ανάπτυξη νέων εργαλείων, καθώς και στην ιστοσελίδα του Bootstrap: https://getbootstrap.com/.

Η δομή του Bootstrap είναι σπονδυλωτή αφού αποτελείται από ένα σύνολο διαφορετικών stylesheets CSS σε συγκεκριμένη σειρά και διάταξη που καλούνται να εφαρμόσουν τα συστατικά που χρειαζόμαστε από το πακέτο εργαλείων. Η δομή του κεντρικού φακέλου του Framework που θα κατεβάσουμε από την ιστοσελίδα για να χρησιμοποιήσουμε στα έργα μας παρουσιάζεται στον πίνακα 4:

```
bootstrap/
├── dist/
│   ├── css/
│   └── js/
├── site/
│   └──content/
│       └── docs/
│           └── 5.0/
│               └── examples/
├── js/
└── scss/
```

Πίνακας 4 Η Δομή της ρίζας – φακέλου του Framework Bootstrap

Στον παρακάτω πίνακα γίνεται μια εμβάθυνση στα αρχεία stylesheet CSS που περιλαμβάνει το Bootstrap καθώς και στα αρχεία Javascript.

```
bootstrap/
├── css/
│   ├── bootstrap-grid.css
│   ├── bootstrap-grid.css.map
│   ├── bootstrap-grid.min.css
│   ├── bootstrap-grid.min.css.map
│   ├── bootstrap-grid.rtl.css
│   ├── bootstrap-grid.rtl.css.map
│   ├── bootstrap-grid.rtl.min.css
```

```
│   ├── bootstrap-grid.rtl.min.css.map
│   ├── bootstrap-reboot.css
│   ├── bootstrap-reboot.css.map
│   ├── bootstrap-reboot.min.css
│   ├── bootstrap-reboot.min.css.map
│   ├── bootstrap-reboot.rtl.css
│   ├── bootstrap-reboot.rtl.css.map
│   ├── bootstrap-reboot.rtl.min.css
│   ├── bootstrap-reboot.rtl.min.css.map
│   ├── bootstrap-utilities.css
│   ├── bootstrap-utilities.css.map
│   ├── bootstrap-utilities.min.css
│   ├── bootstrap-utilities.min.css.map
│   ├── bootstrap-utilities.rtl.css
│   ├── bootstrap-utilities.rtl.css.map
│   ├── bootstrap-utilities.rtl.min.css
│   ├── bootstrap-utilities.rtl.min.css.map
│   ├── bootstrap.css
│   ├── bootstrap.css.map
│   ├── bootstrap.min.css
│   ├── bootstrap.min.css.map
│   ├── bootstrap.rtl.css
│   ├── bootstrap.rtl.css.map
│   ├── bootstrap.rtl.min.css
│   └── bootstrap.rtl.min.css.map
└── js/
    ├── bootstrap.bundle.js
    ├── bootstrap.bundle.js.map
    ├── bootstrap.bundle.min.js
    ├── bootstrap.bundle.min.js.map
    ├── bootstrap.esm.js
    ├── bootstrap.esm.js.map
    ├── bootstrap.esm.min.js
    ├── bootstrap.esm.min.js.map
    ├── bootstrap.js
    ├── bootstrap.js.map
    ├── bootstrap.min.js
    └── bootstrap.min.js.map
```

Πίνακας 5 Το σύνολο των αρχείων CSS & Javascript που περιλαμβάνει το Framework Bootstrap.

Πρέπει να αναφέρουμε ότι το Framework παρέχει επίσης την δυνατότητα προσαρμογής των αρχείων CSS μέσω της χρήσης μεταβλητών, λειτουργιών και operators ή ακόμη και ένθετους φορείς ή αλλιώς μείγματα γνωστά ως mixins.

Μερικά από τα πιο γνωστά πλεονεκτήματα του Bootstrap είναι τα παρακάτω:

- **Σύστημα πλέγματος (Grid System):** Εξ' ορισμού του το Framework έχει ως ελάχιστο πλάτος τα 940pixel. Επειδή όμως πολλές φορές οι προγραμματιστές χρειάζονται πιο

ευέλικτες διατάξεις για τις διαφορετικές διαστάσεις συσκευών, είναι απαραίτητη η ύπαρξη και η χρήση του πλέγματος, ενός στοιχείου που μέσα από τις κατάλληλες κλάσεις μπορεί η εμφάνιση του να αλλάζει ανάλογα με τις διαστάσεις της εκάστοτε συσκευής.

- **Responsive Design – Ανταποκρίσιμος Σχεδιασμός:** Όπως αναφέραμε πιο πάνω ο σχεδιασμός μιας ιστοσελίδας ή μιας εφαρμογής που εμφανίζεται καλαίσθητα και ανταποκρίνεται στις διαφορετικές διαστάσεις είναι ο κύριος στόχος του Bootstrap, δίνοντας την δυνατότητα δυναμικής διάταξης σε όλα τα στοιχεία περιβάλλοντος είτε αυτά είναι εικόνες, πίνακες ή ακόμα και κουμπιά.

- **Επαναχρησιμοποιούμενα συστατικά:** Ένα σημαντικό ζήτημα στην ανάπτυξη ιστοσελίδων είναι η ανάγκη ευέλικτων και επαναχρησιμοποιούμενων blocks κώδικα. Αυτό επιτυγχάνεται στο Bootstrap με την πληθώρα ύπαρξη έτοιμων κλάσεων και συναρτήσεων που επιτελούν τις πιο χρήσιμες λειτουργίες και μπορούν να χρησιμοποιηθούν μέσω ελάχιστων γραμμών κώδικα σε οποιοδήποτε αρχείο ή σημείο αρχείου που έχει προηγουμένως συνδεθεί με το κεντρικό stylesheet CSS αρχείο του Bootstrap.

- **Ενιαία και Σύγχρονη εμφάνιση Ιστοσελίδας:** Αν κάτι ξεχωρίζει το συγκεκριμένο Framework από άλλα παρόμοια αυτό είναι η ομοιόμορφή και σύγχρονη εμφάνιση που προσφέρει στις ιστοσελίδες που κατασκευάζουμε με ην χρήση στοιχείων όπως η σκίαση, τα πλέγματα, οι μοντέρνες γραμματοσειρές. Ακόμα και αν δεν αναπτύξουμε κάποιον δικό μας κώδικα μορφοποίησης, μόνο με την χρήση των CSS αρχείων του Bootstrap θα καταφέρουμε να δημιουργήσουμε μια μοντέρνα ιστοσελίδα πλήρως προσαρμοσμένη στους διαφορετικούς browsers και στις διαστάσεις συσκευών.

- **Javascript Συστατικά:** ένα ακόμη πλεονέκτημα του Bootstrap είναι η δυνατότητα χρήσης έτοιμων Javascript συστατικών για την άμεση δημιουργία στοιχείων όπως το carousel, τα παράθυρα διαλόγου, οι επεξηγήσεις, το scrollspy, τα popovers κ.α. τα οποία για να δημιουργηθούν πολλές φορές απαιτούν την ανάπτυξη μεγάλου σε όγκο κώδικα, πολύωρη εργασία και ένα μεγάλο πλήθος βιβλιοθηκών – libraries. Τα προβλήματα αυτά όμως λύνονται με την χρήση του έτοιμου Javascript κώδικα που παρέχει ελεύθερα το Framework. Πρέπει να τονίσουμε ότι μέχρι και πριν ένα χρόνο τα αρχεία Javascript ήταν σε μορφή JQuery Plugins με την τελευταία έκδοση όμως την v5.0.2 το Bootstrap πλέον έχει σχεδιαστεί για την χρήση αυτούσιου Javascript κώδικα. Όπως πληροφορούμαστε και

από την σελίδα του Framework, εάν θέλουμε με την νέα έκδοση να εξακολουθούμε να χρησιμοποιούμε τα plugins γραμμένα σε JQuery μορφή τότε θα πρέπει το Bootstrap να αναγνωρίσει την λέξη JQuery στο αντικείμενο window ώστε να «φορτώσει» όλα τα συστατικά στο σύστημα JQuery. Έπειτα από αυτό η κλήση για παράδειγμα των tooltips θα γίνεται ως εξής: **('[data-bs-toggle = "tooltip"]'). tooltip ()**. Με τον ίδιο τρόπο θα καλούνται και τα υπόλοιπα συστατικά.

6.2.Εγκατάσταση και Χρήση του Bootstrap

Η εγκατάσταση του Framework είναι μια εύκολη διαδικασία και υλοποιείται με δύο διαφορετικούς τρόπους:

- **Σύνδεση των αρχείων Bootstrap με χρήση εξωτερικών συνδέσμων.**

Στην περίπτωση αυτή δεν χρειάζεται να «κατεβάσουμε» τα αρχεία τοπικά στον υπολογιστή μας και να τα εισάγουμε στον φάκελο του project μας, αλλά μας δίνεται η δυνατότητα να συνδέσουμε το κεντρικό αρχείο CSS του Bootstrap με το αρχείο Html του έργου μας μέσω της εντολής link εισάγοντας έναν εξωτερικό σύνδεσμο που οδηγεί στον server που έχουν αποθηκευτεί τα αρχεία του Framework. Η σύνδεση με την εντολή link συντάσσεται στο head tag του αρχείου Html. Έτσι για το αρχείο CSS του Bootstrap έχουμε την εξής γραμμή κώδικα:

```
<link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.m
in.css" rel="stylesheet" integrity="sha384-
EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC"
crossorigin="anonymous">
```

Για την σύνδεση των Javascript Plugins του Bootstrap που θέλουμε να χρησιμοποιήσουμε, θα εισάγουμε τους εξωτερικούς συνδέσμους με ένα script tag μέσα στο body tag του αρχείου Html ακριβώς πάνω από το κλείσιμο του <body> και με την σειρά που ορίζεται στην ιστοσελίδα του Framework.

Αν θέλουμε να συνδέσουμε όλα τα αρχεία Javascript με μια εντολή σε μορφή bundle τότε θα γράψουμε την εξής εντολή κώδικα:

```
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bun
dle.min.js" integrity="sha384-
MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM"
crossorigin="anonymous"></script>
```

Διαφορετικά αν θέλουμε να εισάγουμε τα αρχεία ξεχωριστά θα πρέπει να προσέξουμε ώστε το αρχείο popper να συνδεθεί πρώτο πάνω από το αρχείο Javascript Bootstrap όπως φαίνεται εδώ:

```
<script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.2/dist/umd/popper.
min.js" integrity="sha384-
IQsoLXl5PILFhosVNubq5LC7Qb9DXgDA9i+tQ8Zj3iwWAwPtgFTxbJ8NT4GN1R8p"
crossorigin="anonymous"></script>


<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.min
.js" integrity="sha384-
cVKIPhGWiC2Al4u+LWgxfKTRIcfu0JTxR+EQDz/bgldoEyl4H0zUF0QKbrJ0EcQF"
crossorigin="anonymous"></script>
```

Ο παραπάνω τρόπος σύνδεσης των αρχείων φαίνεται στον παρακάτω κώδικα ο οποίος αποτελεί το standard – generic Html αρχείο που θα χρησιμοποιήσουμε για την ανάπτυξη ιστοσελίδας με τον συγκεκριμένο τρόπο σύνδεσης του Bootstrap:

```
<!doctype html>
<html lang="en">
  <head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1">

    <!-- Bootstrap CSS -->
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.m
in.css" rel="stylesheet" integrity="sha384-
EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC"
crossorigin="anonymous">

    <title>Hello, world!</title>
  </head>
  <body>
    <h1>Hello, world!</h1>

    <!-- Optional JavaScript; choose one of the two! -->

    <!-- Option 1: Bootstrap Bundle with Popper -->
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bun
dle.min.js" integrity="sha384-
MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM"
```

```
crossorigin="anonymous"></script>

    <!-- Option 2: Separate Popper and Bootstrap JS -->
    <!--
    <script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.2/dist/umd/popper.
min.js" integrity="sha384-
IQsoLXl5PILFhosVNubq5LC7Qb9DXgDA9i+tQ8Zj3iwWAwPtgFTxbJ8NT4GN1R8p"
crossorigin="anonymous"></script>
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.min
.js" integrity="sha384-
cVKIPhGWiC2Al4u+LWgxfKTRIcfu0JTxR+EQDz/bgldoEyl4H0zUF0QKbrJ0EcQF"
crossorigin="anonymous"></script>
    -->
  </body>
</html>
```

- **Κατέβασμα του φακέλου Bootstrap και εισαγωγή του στο δικό μας έργο**

Η μέθοδος αυτή είναι και η πιο απλή αφού το μόνο που χρειάζεται είναι να «κατεβάσουμε» τον φάκελο του Bootstrap και να τον αποθηκεύσουμε στο φάκελο του project μας. Από εκεί καλούμε τα βασικά αρχεία CSS & Javascript του Bootstrap με τον τρόπο που αναφέραμε πιο πάνω χρησιμοποιώντας τις εντολές link και script με την διαφορά ότι τώρα δεν θα έχουμε εξωτερικούς συνδέσμους αλλά θα πρέπει να προσδιορίσουμε το μονοπάτι των αρχείων καταγράφοντας στις εντολές το path.

Με αυτόν τον τρόπο σύνδεσης μπορούμε να χρησιμοποιήσουμε στον editor που αναπτύσσουμε τον κώδικα κάποιόν από τους πιο γνωστούς package managers. Αυτό σημαίνει ότι έχουμε τις παρακάτω εντολές για εγκατάσταση του πακέτου Bootstrap στους εξής managers:

Εγκατάσταση σε node.js με npm manager:

```
npm install bootstrap
```

Στην περίπτωση χρήσης του node.js ο φάκελος του project μας θα έχει την εξής μορφή μετά την εγκατάσταση του Bootstrap:

```
project-name/
├── build/
├── node_modules/
│   └── bootstrap/
│   └── popper.js/
├── scss/
│   └── custom.scss
├── src/
│   └── index.html
│   └── index.js
└── package.json
```

Πίνακας 6 Η κατανομή των αρχείων στο φάκελο του project μας μετά την εγκατάσταση του Bootstrap σε node.js.

- **Εγκατάσταση σε node.js με yarn manager:**

```
yarn add bootstrap
```

- **Εγκατάσταση σε RubyGems με bundler και rubygems γράφοντας στο gemfile την παρακάτω εντολή:**

```
gem 'bootstrap', '~> 5.0.2'
```

Εάν δεν χρησιμοποιούμε bundler τότε η εντολή αλλάζει σε:

```
gem install bootstrap -v 5.0.2
```

- **Εγκατάσταση και χρήση των αρχείων Bootstrap με Composer:**

```
composer require twbs/bootstrap:5.0.2
```

- **Εγκατάσταση σε .NET και διαχείριση με nuget:**

Για ολόκληρο το πακέτο Bootstrap

```
Install-Package bootstrap
```

Για τα αρχεία SASS Bootstrap

```
Install-Package bootstrap.sass
```

Θα πρέπει να προσθέσουμε ότι στην περίπτωση αυτής της μεθόδου τα αρχεία SASS απαιτούν κατά την χρήση τους ειδικό Compiler σε CSS αρχεία. Στην παρούσα πτυχιακή για την ανάπτυξη του συστήματος ραντεβού για ιατρική επίσκεψη χρησιμοποιήσαμε ως editor το πρόγραμμα visual code με Compiler SASS αρχείων το plugin “Live Sass Compiler”.

6.3.Συμβατότητα του Framework στους Φυλλομετρητές

Όπως αναφέραμε προηγουμένως ένα από τα πλεονεκτήματα του Bootstrap είναι η ευρεία συμβατότητα του Framework με το γενικό σύνολο των φυλλομετρητών. Πάμε να δούμε όμως στον παρακάτω πίνακα την καταγεγραμμένη συμβατότητα τόσο σε κινητές – mobile συσκευές όσο και σε υπολογιστές – desktops.

Mobile Συσκευές

	Chrome	Firefox	Safari	Android Browser & Webview
Android	Υποστηρίζεται	Υποστηρίζεται	-	Υποστηρίζεται από v6.0+
IOS	Υποστηρίζεται	Υποστηρίζεται	Υποστηρίζεται	-

Πίνακας 7 Συμβατότητα του Bootstrap σε mobile συσκευές.

Αξίζει να σημειωθεί ότι σε mobile συσκευές οι proxy browsers όπως οι Opera Mini, Opera Mobile's Turbo mode, UC Browser Mini, Amazon Silk δεν υποστηρίζονται, αν θέλουμε όμως να υπερασπιστούμε το Bootstrap σε αυτήν την έλλειψη το ποσοστό των χρηστών που χρησιμοποιούν τους παραπάνω browsers είναι μόλις ~ 2,16% - 5,79%.

Desktop Συσκευές

	Chrome	Firefox	Microsoft Edge	Opera	Safari
MAC	Υποστηρίζεται	Υποστηρίζεται	Υποστηρίζεται	Υποστηρίζεται	Υποστηρίζεται
Windows	Υποστηρίζεται	Υποστηρίζεται	Υποστηρίζεται	Υποστηρίζεται	-

Πίνακας 8 Συμβατότητα του Bootstrap σε desktop συσκευές.

Παρότι δεν αναφέρεται αν υποστηρίζονται οι φυλλομετρητές Chromium, Chrome για Linux και Firefox για Linux ανεπίσημα κατά γενική ομολογία ιστοσελίδες με Bootstrap εμφανίζονται με σωστή διάταξη και ελάχιστη έλλειψη στοιχείων. Ένα ακόμη σημαντικό στοιχείο που πρέπει να αναφέρουμε είναι η μη συμβατότητα του Internet Explorer σε παλαιότερες εκδόσεις του Bootstrap από την v4.

6.4.Η προσβασιμότητα του Bootstrap για άτομα με ειδικές ανάγκες

Η προσβασιμότητα μιας ιστοσελίδας ή μιας διαδικτυακής εφαρμογής από άτομα με ειδικές ανάγκες πρέπει να αποτελεί βασικό κριτήριο για την ανάπτυξη του απαιτούμενου κώδικα. Ο προγραμματιστής που αναλαμβάνει την δημιουργία μιας ιστοσελίδας που θα πληροί τις προϋποθέσεις που απαιτούνται για έναν κατάλληλα σχεδιασμένο ιστότοπο θα πρέπει να χρησιμοποιήσει τις απαραίτητες βιβλιοθήκες κώδικα, προσαρμοσμένα plugins καθώς και ένα ολοκληρωμένο γραφικό σχέδιο. Το Bootstrap έρχεται να λύσει αν όχι όλα τα προβλήματα που δημιουργούνται κατά την διάρκεια σχεδιασμού μιας τέτοιας ιστοσελίδας – εφαρμογής, έστω κάποια από αυτά παρέχοντας βασικά εργαλεία και στοιχεία για την βέλτιστη εμφάνιση και προσαρμοστικότητα της ιστοσελίδας σε βαθμό που ακόμα και αν χρησιμοποιήσουμε μόνο το Framework είναι ικανό να κατασκευάσουμε ένα ιστότοπο που ακολουθεί τις προϋποθέσεις του πρωτοκόλλου WSAG 2.1. Το πρωτόκολλο αυτό αφορά τα πρότυπο στοιχεία που απαιτούνται για την ευκολότερη πρόσβαση των ατόμων με ειδικές ανάγκες.

Το Bootstrap καταφέρνει να συνδράμει σε κάποια από τα παρακάτω στοιχεία προσβασιμότητας:

- **Δομική Σήμανση – Structural Markup:** Αποτελείται από μια σειρά από εντολές και στοιχεία που βοηθούν στην βέλτιστη κατασκευή ενός εύχρηστου και πρακτικού layout μιας διαδικτυακής σελίδας
- **Διαδραστικά Στοιχεία – Interactive Components:** Περιλαμβάνουν τα tooltips, dropdown menus, modals τα οποία με την σειρά τους λειτουργούν βάσει της κίνησης και των ενεργειών του ποντικιού. Στην περίπτωση των σελίδων που απευθύνονται σε άτομα με ειδικές ανάγκες θα πρέπει αυτά τα στοιχεία να παραμένουν λειτουργικά χωρίς την χρήση του ποντικιού αλλά με την λειτουργία των screen readers.
- **Χρωματική Αντίθεση – Color Contrast:** Το Bootstrap από την δημιουργία του χρησιμοποιεί για την κατασκευή generic σελίδων μια παλέτα χρωμάτων την οποία μας δίνεται η δυνατότητα να την μεταβάλλουμε ανάλογα με τις απαιτήσεις μας. Σύμφωνα με το πρωτόκολλο WSAG μια σελίδα απαιτείται να διαθέτει μια επιπλέον χρωματική παλέτα αποτελούμενη από τα αντίθετα χρώματα της πρώτης παλέτας που έχουμε ορίσει. Αυτό γίνεται γιατί θα πρέπει να υπάρχει μια ευκρινής αντίθεση των κειμένων και των εικόνων από το υπόλοιπο layout της σελίδας έτσι ώστε άτομα με μειωμένη όραση να μπορούν ξεχωρίσουν τα σημαντικά στοιχεία στην σελίδα.

- **Οπτικά Κρυφό Περιεχόμενο – Optical Hidden Content:** Μια επιπλέον λειτουργία έρχεται να προστεθεί για να βοηθήσει ανθρώπους με προβλήματα όρασης. Αυτή είναι κλάση .visually-hidden η οποία χρησιμοποιείται για να ‘κρύψει’ οποιοδήποτε στοιχείο θέλουμε από την σελίδα. Η λειτουργία αυτή είναι πολύ χρήσιμη στις συσκευές screen readers καθώς κρύβει οπτικά χρωματικά στοιχεία και λεπτομέρειες που δεν είναι απαραίτητα και σημαντικά για την λειτουργικότητα της σελίδας από χρήστες χωρίς οπτική επαφή. Θα πρέπει να σημειωθεί ότι για διαδραστικά στοιχεία όπως τα γνωστά skip links ή κουμπιά που θέλουμε να κρυφτούν αλλά να εμφανίζονται μόνο όταν ο χρήστης θέλει να τα χρησιμοποιήσει, υπάρχει η κλάση .visually-hidden-focusable που επιτελεί ακριβώς αυτήν την λειτουργία, δηλαδή κρύβει τα στοιχεία μέχρι ο χρήστης να αλληλοεπιδράσει μαζί τους.
- **Μειωμένη Κίνηση – Reduced Motion:** Η λειτουργία αυτή έρχεται να λύσει προβλήματα χρήσης της σελίδας από άτομα κυρίως με προβλήματα επιληπτικών κρίσεων. Σε μια σελίδα μπορεί να υπάρχουν στοιχεία και περιεχόμενο με κινητικότητα για παράδειγμα animated γραφικά, gifs, transitions μεταξύ των στοιχείων. Για να μην συμβάλλει η έντονη κινητικότητα στο πρόβλημα της επιληψίας το Bootstrap φέρνει την λειτουργία **prefers-reduced-motion media feature**, η οποία με την σειρά της αναλαμβάνει να μειώσει την κινητικότητα σε επιλογές transition, modal open-close movements, carousel sliding κ.α.. Σε browsers που υποστηρίζουν την λειτουργία αυτή αλλά ο χρήστης δεν έχει δηλώσει ακόμα την επιλογή του στην μειωμένη κίνηση μέσω της εντολής prefers-reduced-motion, το Bootstrap θεωρεί ως προεπιλογή για την συγκεκριμένη εντολή το no-preference (**prefers-reduced-motion: no-preference**) και επιπλέον εισάγει στην σελίδα την λειτουργία **smooth-scrolling** που είναι υπεύθυνη για την αργή μετακύληση της σελίδας κατά την διάρκεια του scroll.

6.5.Σύστημα RTL

Το σύστημα RTL (Right to Left) χρησιμοποιείται για να βοηθήσει την μετατροπή της σελίδας από τα δεξιά στα αριστερά για την βέλτιστη κατασκευή των σελίδων που προορίζονται για χρήστες χωρών που χρησιμοποιούν το σύστημα γραφής και ανάγνωσης κειμένων από τα δεξιά προς τα αριστερά σε αντίθεση με τους χρήστης των δυτικών χωρών.

Το Bootstrap έχει δώσει λύση για την γρήγορη μετατροπή της σελίδας σε RTL με την χρήση ενός CSS αρχείου που έχει δημιουργήσει για την μορφοποίηση των στοιχείων σε RTL αφού πρώτα

προβούμε σε μια πολύ σημαντική προσθήκη η οποία περιλαμβάνει την χρήση των attributes lang και dir στην εντολή <html>. Με την χρήση των δύο αυτών στοιχείων έχουμε το παρακάτω tag:

```
<html lang="ar" dir="rtl">
```

Με το attribute lang υποδηλώνουμε την γλώσσα στην οποία θα γραφτούν τα κείμενα του html αρχείου, στην περίπτωση μας έχουμε δηλώσει το 'ar' που πολύ απλά σημαίνει ότι θα γράψουμε τα κείμενα στην αραβική γλώσσα. Με το attribute dir δηλώνουμε την διεύθυνση του κειμένου (direction) δηλαδή στην παραπάνω περίπτωση την RTL από τα δεξιά προς τα αριστερά.

Η παρακάτω εντολή χρησιμοποιείται για να συνδέσει το html αρχείο με το αντίστοιχο CSS αρχείο που αναφέραμε πιο πάνω και προορίζεται για την μορφοποίηση των στοιχείων σε γραφή RTL:

```
<link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.0/dist/css/bootstrap.r
tl.min.css" integrity="sha384-
XfhC/Sid4FIGSXYebcOtcSCRFkd/zZzAMVipf0bNWucloRvcKK2/dpVWodQbQ1Ek"
crossorigin="anonymous">
```

Επιπρόσθετα θα πρέπει να σημειωθεί ότι δίνεται η δυνατότητα να εισάγουμε διαφορετικές επιλογές μορφοποίησης στο αρχείο CSS που θα κατασκευάσουμε ανάλογα με την διεύθυνση του κειμένου για παράδειγμα μπορούμε να εισάγουμε διαφορετικό πάχος γραμματοσειράς για κείμενο LTR ή RTL στην ίδια εντολή εισάγοντας το πάχος της γραμματοσειράς σε RTL μέσα σε αγκύλες και συνοδευόμενες από τα σύμβολα των σχολίων. Μπροστά από την πρώτη αγκύλη τοποθετείται μια δίεση ώστε να υποδηλώνεται ότι οι συγκεκριμένες ρυθμίσεις δεν είναι σχόλια αλλά θα πρέπει να ληφθούν σοβαρά υπόψιν από τον compiler όταν το αρχείο Html αλλάζει διεύθυνση κειμένου.

```
$font-weight-bold: 700 #{/* rtl:600 */} !default;
```

Το attribute !default υποδηλώνει ότι το πάχος **700** της γραμματοσειράς είναι η προεπιλογή και χρησιμοποιείται στο αρχείο **bootstrap.css** ενώ το πάχος **600** είναι για γραφή σε RTL και θα εισαχθεί στο αρχείο **bootstrap.rtl.css**.

Στην περίπτωση που θέλουμε να χρησιμοποιήσουμε στο ίδιο αρχείο Html και τα δύο συστήματα LTR & RTL τότε θα πρέπει να χρησιμοποιήσουμε ένα string map. Το στοιχείο αυτό είναι ένα array δηλαδή πίνακας αντικειμένων χάρτη, όπου κάθε αντικείμενο ορίζει μια αντιστοίχιση μεταξύ συμβολοσειρών διεύθυνσης. Στην ουσία με τον παρακάτω κώδικα δηλώνουμε στον compiler την αλλαγή και μετατροπή των στοιχείων της σελίδας από το ένα σύστημα στο άλλο ανάλογα με την διεύθυνση γραφής των κειμένων. Ο compiler «βλέπει» το κείμενο σε LTR γραφή και το

αντιστοιχεί με το κατάλληλο CSS αρχείου του Bootstrap, αν όμως «δει» ότι τα κείμενα παρακάτω αλλάζουν σε γραφή RTL τότε θα αναγκαστεί να αλλάξει και να χρησιμοποιήσει το αντίστοιχο αρχείο CSS που έχει δημιουργηθεί για την μορφοποίηση της RTL.

```
/* rtl:begin:options: {
  "autoRename": true,
  "stringMap":[ {
    "name": "ltr-rtl",
    "priority": 100,
    "search": ["ltr"],
    "replace": ["rtl"],
    "options": {
      "scope": "*",
      "ignoreCase": false
    }
  } ]
} */
.ltr {
  @import "../node_modules/bootstrap/scss/bootstrap";
}
/*rtl:end:options*/
```

6.6. Το πλέγμα - Grid

Ένα από τα σημαντικότερα στοιχεία του Bootstrap είναι το πλέγμα – Grid. Ουσιαστικά μας δίνεται η δυνατότητα να κατασκευάσουμε έναν νοητό πίνακα με γραμμές και στήλες και στα κελιά στα οποία θα δημιουργηθούν να εισάγουμε οποιοδήποτε στοιχείο θέλουμε όπως εικόνα, κείμενο, video κ.α..

Το πλέγμα κατασκευάζεται με την βοήθεια εμφωλευμένων tags div. Οι γραμμές δηλώνονται με την κλάση row και οι στήλες με την κλάση col και τα παράγωγα της. Να σημειωθεί ότι όταν χρησιμοποιούμε το πλέγμα τότε χωρίζεται το layout σε 12 νοητές στήλες. Ενώ μπορούμε να έχουμε όσες γραμμές θέλουμε στο πλέγμα, οι στήλες που μπορούμε να δημιουργήσουμε σε κάθε γραμμή δεν μπορούν να υπερβαίνουν στο άθροισμα τους τις 12. Έτσι το bootstrap έχει δημιουργήσει όπως αναφέραμε και πιο πάνω κάποια παράγωγα της κλάσης col με σκοπό τον διαχωρισμό και την εύκολη καταμέτρηση των στηλών άσχετα σε ποια συσκευή προβάλλεται η σελίδα.

Οι δώδεκα στήλες μέσα σε μια γραμμή ορίζονται χρησιμοποιώντας απλά την κλάση col:

```
<div class="container">
  <div class="row">
```

```
    <div class="col"> Column 1 </div>
    <div class="col"> Column 2 </div>
    <div class="col"> Column 3 </div>
    <div class="col"> Column 4 </div>
    <div class="col"> Column 5 </div>
    <div class="col"> Column 6 </div>
    <div class="col"> Column 7 </div>
    <div class="col"> Column 8 </div>
    <div class="col"> Column 9 </div>
    <div class="col"> Column 10 </div>
    <div class="col"> Column 11 </div>
    <div class="col"> Column 12 </div>
  </div>
</div>
```

Στην περίπτωση που δεν θέλουμε να έχουμε 12 στήλες σε μία γραμμή αλλά για παράδειγμα μόνο έξι τότε θα πρέπει να χρησιμοποιήσουμε τα παράγωγα μεγέθους. Στην ουσία προσθέτουμε μια παύλα και αριθμό δίπλα από την κλάση col δηλώνοντας έτσι το μέγεθος της στήλης ή αλλιώς τον χώρο που θέλουμε να καταλάβει η στήλη μέσα στην γραμμή. Έτσι λοιπόν για έξι στήλες θα κατασκευάσουμε έξι divs και θα χρησιμοποιήσουμε το παράγωγο col-2, δηλώνοντας ότι η κάθε στήλη θα καταλαμβάνει χώρο για δύο στήλες.

Προσοχή στον αριθμό των divs που θα κατασκευάσουμε γιατί θα πρέπει να είναι ένας αριθμός όπου το άθροισμα των παραγώγων να μας δίνει το 12. Στην περίπτωση μας έχοντας 6 divs με παράγωγο το 2 επιτυγχάνουμε τον αριθμό 12, αφού 6*2=12 ή col-2 + col-2 + … + col-2 = 12.

```
<div class="container">
  <div class="row">
    <div class="col-2"> Column 1 </div>
    <div class="col-2"> Column 2 </div>
    <div class="col-2"> Column 3 </div>
    <div class="col-2"> Column 4 </div>
    <div class="col-2"> Column 5 </div>
    <div class="col-2"> Column 6 </div>
  </div>
</div>
```

Ένα ακόμα παράδειγμα είναι η δημιουργία δύο στηλών με ανόμοια μεγέθη. Αν δηλαδή θέλουμε η πρώτη στήλη να είναι μεγαλύτερη από την δεύτερη τότε απλά αλλάζουμε τον αριθμό στο παράγωγο προσέχοντας πάντα το άθροισμα των παραγώγων να μας δίνει τον αριθμό 12. Έτσι αν θέλουμε η πρώτη στήλη να καταλαμβάνει 7 θέσεις τότε το παράγωγο της δεύτερης θα είναι 5 γιατί 12 – 7 = 5.

```
<div class="container">
  <div class="row">
    <div class="col-7"> Column 1 </div>
    <div class="col-5"> Column 2 </div>
  </div>
</div>
```

Αξίζει να σημειωθεί ότι άσχετα με το πόσες στήλες έχουμε σε κάθε γραμμή το Bootstrap μας δίνει την δυνατότητα να δημιουργήσουμε ένα responsive πλέγμα όπου θα εμφανίζεται ομοιόμορφα σε κάθε συσκευή. Αυτό το επιτυγχάνουμε χρησιμοποιώντας τα παράγωγα της κλάσης col που αφορούν τα breakpoints, δηλαδή τα σημεία μήκους της οθόνης στα οποία αλλάζει το μέγεθος της συσκευής. Εξ ορισμού το Bootstrap διαθέτει συγκεκριμένα όρια – breakpoints, για παράδειγμα κάτω από τα 320pixel μήκος (width) η μορφοποίηση των στοιχείων γίνεται για συσκευές κινητού. Από 320pixel έως 720pixel αναφερόμαστε στις συσκευές tablet. Ακολουθώντας λοιπόν αυτήν την λογική θα δούμε ότι ένα πλέγμα που κάθε του γραμμή αποτελείται από 6 στήλες μπορεί να φαίνεται ομοιόμορφο και εύχρηστο όταν η σελίδα εμφανίζεται σε οθόνη desktop, όταν όμως το ίδιο πλέγμα παρουσιάζεται σε οθόνη κινητού παρατηρούμε ότι το μήκος των στηλών μικραίνει υπερβολικά σε τέτοιο βαθμό όπου τα στοιχεία των στηλών να συμπιέζονται και να βγαίνουν έξω από τα όρια του πλέγματος. Κάτι τέτοιο δεν είναι πολύ πρακτικό αλλά και ευπαρουσίαστο στα μάτια ενός χρήστη. Αυτό το πρόβλημα έρχονται να λύσουν τα παράγωγα οθόνης. Στην ουσία εισάγουμε μετά την λέξη col και πριν το μέγεθος της στήλης ένα διακριτικό που υποδηλώνει το μέγεθος της οθόνης στην οποία θέλουμε να εφαρμοστεί το συγκεκριμένο μέγεθος στήλης που επιθυμούμε.

Τα διακριτικά οθόνης είναι τα εξής:

```
$grid-breakpoints: (
  xs: 0,      /* Οθόνη με μήκος < 576pixel */
  sm: 576px,  /* Οθόνη με μήκος >= 576pixel < 768pixel */
  md: 768px,  /* Οθόνη με μήκος >= 768pixel < 992pixel */
  lg: 992px,  /* Οθόνη με μήκος >= 992pixel < 1200pixel */
  xl: 1200px, /* Οθόνη με μήκος >= 1200pixel < 1400pixel*/
  xxl: 1400px /* Οθόνη με μήκος >= 1400pixel */
);
```

Γνωρίζοντας πλέον τα διακριτικά μπορούμε να τα τοποθετήσουμε μέσα στην κλάση col. Έτσι θα έχουμε:

```
<div class="container">
  <div class="row">
    <div class="col-12 col-sm-12 col-md-6 col-lg-7 col-xl-7 col-xxl-7">
Column 1 </div>
```

```
    <div class="col-12 col-sm-12 col-md-6 col-lg-5 col-xl-5 col-xxl-5">
Column 2 </div>
  </div>
</div>
```

Με τα παραπάνω παράγωγα καταλαβαίνουμε ότι έχουμε ορίσει διαφορετικό μέγεθος στήλης για κάθε διαφορετικό μέγεθος οθόνης. Δηλαδή σε όλες τις οθόνες που έχουν μήκος >= 992pixel η πρώτη στήλη θα έχει μέγεθος 7 ενώ η δεύτερη 5 αυτό το διακρίνουμε στο μέγεθος στήλης που έχουμε εισάγει δίπλα από τα διακριτικά lg, xl, xxl. Σε οθόνες με μήκος >= 768pixel και < 992pixel το μέγεθος και των δύο στηλών θα είναι 6 άρα η γραμμή του πλέγματος χωρίζεται σε δυο ισόποσες στήλες.

Τέλος συσκευές με μήκος οθόνης < 576pixel δηλαδή mobile συσκευές οι στήλες θα έχουν μέγεθος 12 δηλαδή η κάθε στήλη θα καταλαμβάνει ολόκληρο τον χώρο της γραμμής, έτσι η δεύτερη στήλη θα εμφανίζεται κάτω από την δεύτερη αφού δεν θα χωρά δίπλα της σχηματίζοντας δυο νοητές γραμμές πλέγματος.

Προσοχή: Όταν χρησιμοποιούμε τα παράγωγα οθόνης τότε το απλό παράγωγο col-* χωρίς διακριτικό οθόνης αναφέρεται σύμφωνα με την νέα έκδοση του Bootstrap στις πολύ μικρότερες σε μήκος οθόνες ακριβώς όπως και το παράγωγο col-sm-*.

7. ΑΝΑΠΤΥΞΗ ΣΥΣΤΗΜΑΤΟΣ ΚΑΤΑΓΡΑΦΗΣ ΡΑΝΤΕΒΟΥ ΓΙΑ ΙΑΤΡΙΚΗ ΕΠΙΣΚΕΨΗ

Στο κεφάλαιο αυτό θα προβούμε στην λεπτομερή παρουσίαση και επεξήγηση του συστήματος που αναπτύξαμε για την καταγραφή ηλεκτρονικών ραντεβού για ιατρική επίσκεψη.

7.1.Περιεχόμενα φακέλου του Project

Στην παρακάτω λίστα απαριθμούνται τα διαφορετικά αρχεία που δημιουργήθηκαν για την λειτουργία του συστήματος και αναγράφεται επιγραμματικά ο σκοπός της ύπαρξης τους και ο τρόπος λειτουργίας τους.

Φάκελος images:

Στον φάκελο αυτό υπάρχουν εικόνες που έχουν χρησιμοποιηθεί για τα προφίλ των χρηστών του συστήματος που έχουμε εισάγει στο σύστημα και στην βάση δεδομένων για να υπάρχει ένα ενδεικτικό υλικό κατά την διάρκεια της παρουσίασης του συστήματος. Περιλαμβάνει και τον

δεύτερο φάκελο SVG μέσα στον οποίο υπάρχουν πρότυπα αρχεία σε μορφή SVG τα οποία χρησιμοποιούνται ως εικόνα του νέου μέλους που εγγράφεται στο σύστημα.

Φάκελος fonts:

Περιέχει τα αρχεία των γραμματοσειρών που έχουμε χρησιμοποιήσει στην πλατφόρμα. Στην παρούσα εργασία έχει χρησιμοποιηθεί η γραμματοσειρά Roboto με σύνδεση import μέσα στο κεντρικό particle αρχείο **_global.scss**, ενώ ο ορισμός των απαραίτητων κλάσεων μορφοποίησης των κειμένων βρίσκεται στο αρχείο **_typography.scss**.

Φάκελος SCSS:

Περιέχει όλα τα αρχεία της γραφικής μορφοποίησης σε μορφή SCSS που όπως προαναφέραμε είναι μια πιο εύχρηστη και απλουστευμένη μορφή εμφωλευμένου κώδικα CSS το οποίο πρέπει με την χρήση ειδικών εργαλείων να μεταγλωττιστεί σε ένα γενικό αρχείο CSS. Το μεταγλωττισμένο αρχείο CSS είναι αποθηκευμένο στον αντίστοιχο φάκελο CSS όπως θα δούμε παρακάτω. Μέσα στον φάκελο SCSS περιέχονται το κυρίως αρχείο styles.scss από το οποίο καλούνται τα particles αρχεία που χρησιμοποιούνται για τον καταμερισμό της μορφοποίησης.

Φάκελος CSS:

Στον φάκελο αυτό έχουμε αποθηκεύσει το κυρίως αρχείο styles.css που μεταγλωττίζεται από το αντίστοιχο αρχείο scss που αναφέραμε προηγουμένως. Το αρχείο αυτό περιλαμβάνει όλο τον κώδικα μορφοποίησης που έχουμε αναπτύξει για την πλατφόρμα και ο οποίος περιέχει γραφικά στοιχεία που δεν υπόκεινται στο Framework Bootstrap.

Φάκελος JS:

Περιέχει τα παρακάτω αρχεία Javascript:

- **signup.js**
- **login.js**
- **edit_admin_profile.js**
- **edit_doctor_profile.js**
- **edit_patient_profile.js**
- **admin_appointment_list.js**

- **admin_dash_doctors_table.js**
- **admin_dash_patients_table.js**
- **doctor_working_hours_list.js**
- **patient_appointment_list.js**
- **patient_book_apppointment.js**

Φάκελος Vendors:

Όπως λέει και το όνομα του σχετίζεται με τους Vendors, δηλαδή τους «προμηθευτές ή παρόχους» έτοιμου κώδικα όπως για παράδειγμα το Framework Bootstrap που έχουμε προαναφέρει. Το Bootstrap μας παρέχει δωρεάν κώδικα CSS και JavaScript για να μπορούμε πιο εύκολα και γρήγορα να αναπτύσσουμε οποιαδήποτε ιστοσελίδα, ηλεκτρονική πλατφόρμα ή εφαρμογή θέλουμε. Στον φάκελο λοιπόν Vendor υπάρχουν οι παρακάτω φάκελοι που περιέχουν αρχεία CSS και JavaScript του Bootstrap:

- **Datatables**
- **Datatables-1.11.5**
- **JQuery Easing**
- **Parsley**
- **UxSolutions Bootstrap Datepicker**

Φάκελος Classes:

Αποτελείται από τα εξής αρχεία:

- **Appointment.php:** Είναι το αρχείο στο οποίο ορίζεται:
    - η σύνδεση με την βάση δεδομένων που έχουμε κατασκευάσει,
    - η δήλωση του constructor του αντικειμένου «ραντεβού» καθώς και των μεθόδων του αντικειμένου οι οποίες αποτελούνται στην ουσία από τα ερωτήματα SQL που χρησιμοποιούνται για να εμφανίσουν τα στοιχεία των εγγραφών που θέλουμε.

— **class.phpmailer.php:** Το αρχείο αυτό χρησιμοποιείται όταν προσπαθεί ένας νέος ασθενής να εγγραφεί στο σύστημα. Στην φόρμα εγγραφής εισάγει τα στοιχεία του και όταν πατήσει το κουμπί “register” τότε το αρχείο αυτό καλείται και κατασκευάζεται ένα αντικείμενο “PHPMailer” το οποίο μέσω των μεθόδων του αναλαμβάνει να στείλει μήνυμα επιβεβαίωσης στην διεύθυνση Email που καταχώρησε ο χρήστης στην φόρμα εγγραφής. Είναι το γνωστό Email Verification μέσω του οποίου καλείται ο νέος χρήστης να επιβεβαιώσει την πιστότητα των στοιχείων του πατώντας το κρυπτογραφημένο link που θα σταλεί στο Email του από τον PHPMailer.

— **class.smtp.php:** Ορίζεται ο constructor του αντικειμένου SMTP, στην ουσία δηλαδή το αντικείμενο αυτό καλείται και κατασκευάζεται ώστε να δημιουργηθεί μια ασφαλής σύνδεση στον Server για την αποστολή του Email επιβεβαίωσης που στέλνει στον χρήστη ο PHPMailer.

— **pdf.php:** Είναι ο constructor του αντικειμένου pdf ο οποίος καλείται για να κατασκευάσει ένα αντικείμενο pdf κάθε φορά που ένας ασθενής θέλει να «κατεβάσει» τα στοιχεία του ραντεβού του σε pdf.

— **Φάκελος dompdf:** Ο φάκελος αυτός εμπεριέχει τα αρχεία που απαιτούνται για να μετατρέψει τα στοιχεία του εκάστοτε ραντεβού που αντλούνται από την βάση δεδομένων σε pdf.

— **Φάκελος actions:** Περιλαμβάνει τα αρχεία που αποτελούνται από τον κατάλληλο κώδικα php για τον ορισμό των διαφόρων εργασιών που μπορεί να εκτελέσει ένας χρήστης της πλατφόρμας.

Φάκελος admin: Αποτελείται από τα αρχεία του χρήστη admin. Κάθε αρχείο είναι και μια διαφορετική ιστοσελίδα ανάλογα με το μενού του διαχειριστή της πλατφόρμας.

Φάκελος doctor: Εμπεριέχει τα αρχεία του χρήστη doctor.

Φάκελος patient: Περιλαμβάνει τα αρχεία του χρήστη patient.

Αρχεία στην ρίζα – root folder του project: Τα παρακάτω αρχεία βρίσκονται μέσα στον κεντρικό φάκελο του project και αυτό γιατί είναι κοινά για κάθε διαφορετικό χρήστη της πλατφόρμας, επομένως θα πρέπει να είναι και εύκολα προσβάσιμα:

- Index.php
- Login.php
- Signup.php
- Logout.php
- Download.php
- Verify.php

Μετά την επιγραμματική αναφορά στα αρχεία του project θα αναλύσουμε στα παρακάτω κεφάλαια την κυρίως λειτουργία τους, την μεταξύ τους σύνδεση καθώς και τον τρόπο κατασκευής τους.

7.2.Μεθοδολογία ανάπτυξης της πλατφόρμας

Στο κεφάλαιο αυτό θα αναλυθεί σε βάθος η μεθοδολογία που χρησιμοποιήθηκε για την κατασκευή της πλατφόρμας όσον αφορά την σύνταξη των αρχείων, την δημιουργία της βάσης δεδομένων και της επικοινωνίας μεταξύ τους.

7.2.1. Η βάση δεδομένων του συστήματος

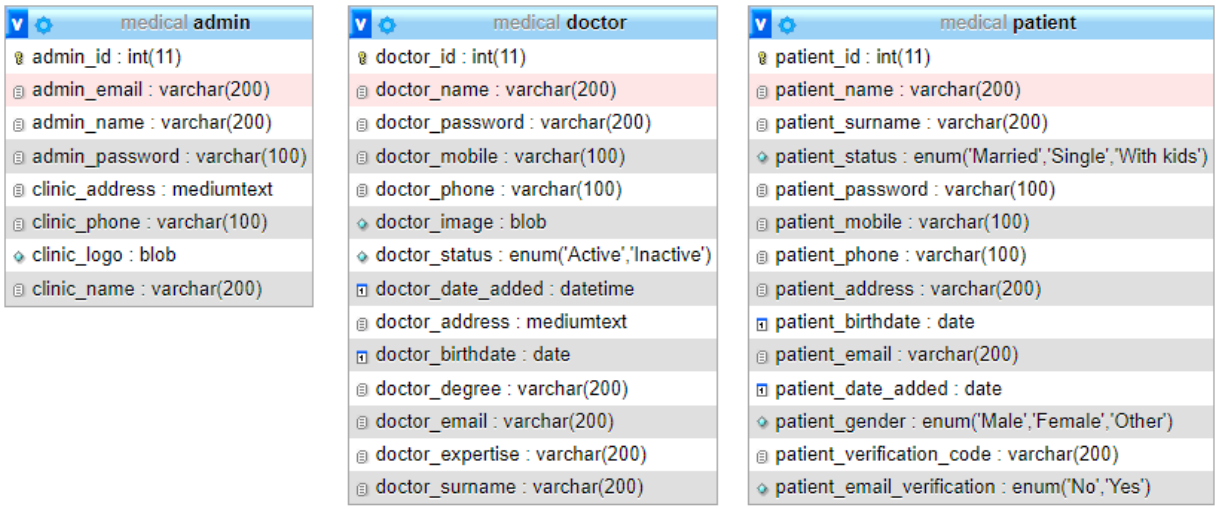
Η βάση δεδομένων του συστήματος είναι ο κεντρικός πυλώνας της πλατφόρμας και αυτό γιατί εκεί αποθηκεύονται όλες οι πληροφορίες που σχετίζονται με την εγγραφή και την σύνδεση των χρηστών αλλά και την καταχώρηση των αιτημάτων – ραντεβού.

7.2.1.1. Σύνθεση της βάσης δεδομένων

Η σύνθεση της βάσης δεδομένων καθορίστηκε βασιζόμενη σε απλά ερωτήματα που θέσαμε θέλοντας να κατανοήσουμε τα δεδομένα και τα ζητούμενα που απαιτούνταν για την δημιουργία μιας ευέλικτης και διαχειρίσιμης πλατφόρμας.

Το πρώτο από τα ερωτήματα που καλούμαστε να απαντήσουμε είναι ποιος θα χρησιμοποιεί την πλατφόρμα. Στην δική μας περίπτωση καθορίσαμε τρία είδη χρηστών, τον διαχειριστή, τον γιατρό και τον ασθενή.

Τα τρία αυτά διαφορετικά είδη χρηστών αποτελούν και τρείς διαφορετικούς πίνακες στην βάση δεδομένων με τα δικά του στοιχεία ο καθένας.



Εικόνα 6 Πίνακες admin, doctor & patient της βάσης δεδομένων medical

Έχοντας καθορίσει πλέον τους χρήστες του συστήματος το επόμενο βήμα είναι να προσδιορίσουμε τις ενέργειες στις οποίες θα μπορούν να προβούν ξεχωριστά ο κάθε ένας ανάλογα με τον ρόλο του ή ακόμα και τις κοινές εργασίες που μπορούν να εκτελέσουν.

Βασική λειτουργία στο σύστημα είναι η εγγραφή νέων χρηστών. Επιλέξαμε να μπορούν να εγγραφούν μέσω ειδικής φόρμας μόνο οι χρήστες που είναι ασθενείς. Εξ ορισμού ο διαχειριστής της πλατφόρμας δεν χρειάζεται να εγγραφεί στην πλατφόρμα αφού θεωρούμε ότι τον ρόλο αυτό θα τον έχει κάποιο στέλεχος της κλινικής. Ο χρήστης γιατρός από την άλλη δεν μπορεί να εγγραφεί έτσι απλά γιατί θα πρέπει να υπάρξει ειδική συνεργασία μεταξύ της κλινικής και των ιατρών, επομένως θα καταχωρούνται στο σύστημα από τον διαχειριστή μετά από απαίτηση της διοίκησης της κλινικής. Έτσι λοιπόν καταλαβαίνουμε ότι ο χρήστης ασθενής είναι ο μόνος ρόλος που απαιτεί την εγγραφή στο σύστημα με σκοπό την επικοινωνία με τους γιατρούς και την καταχώρηση του ραντεβού τους.

Η δεύτερη βασική ενέργεια είναι η σύνδεση των χρηστών η οποία αποφασίσαμε να πραγματοποιείται μέσα από μια κοινή φόρμα για όλα τα είδη των χρηστών. Η σύνδεση γίνεται από όλους τους χρήστες με τις ηλεκτρονικές διευθύνσεις και τους κωδικούς που έχουν καταχωρηθεί στο σύστημα είτε από τους ίδιους κατά την εγγραφή τους είτε από τον διαχειριστή. Την σύνδεση ακολουθεί η αντίστοιχη κοινή λειτουργία η αποσύνδεση.

Έχοντας καθορίσει τις βασικές ενέργειες μεταξύ των χρηστών προχωρούμε στην επιμέρους ανάλυση των λειτουργιών του κάθε χρήστη ξεχωριστά.

Για τον διαχειριστή του συστήματος ορίζουμε ως λειτουργίες που μπορεί να επιτελέσει:

- την καταχώρηση νέων ιατρών στο σύστημα,
- την διαγραφή συγκεκριμένων ιατρών και την αλλαγή των στοιχείων τους,
- την μεταβολή των στοιχείων του προφίλ της κλινικής,
- την επίβλεψη της λίστας των ασθενών και
- την παρακολούθηση της λίστας των ραντεβού.

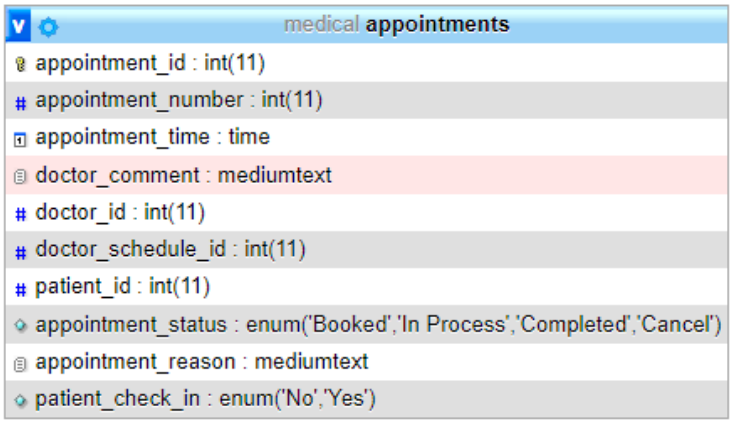
Ένας ιατρός που έχει καταχωρηθεί από τον διαχειριστή, μετά την σύνδεση του στην πλατφόρμα έχει την δυνατότητα:

- να μεταβάλλει τα στοιχεία του στο προφίλ του,
- να προσθέσει ένα καινούργιο ωράριο εργασίας,
- να διαγράψει ένα παλιό ωράριο και
- να παρακολουθήσει τις ήδη καταχωρημένες ώρες εργασίας.

Από την μεριά του ο ασθενής μπορεί να προχωρήσει σε:

- μεταβολή των στοιχείων του προφίλ του στην βάση δεδομένων,
- καταχώρηση ενός καινούργιου ραντεβού μέσω της λίστας με το αναρτημένο ωράριο εργασίας των ιατρών
- παρακολούθηση της πορείας των κλεισμένων του ραντεβού
- ακύρωση προκαθορισμένου ραντεβού.

Μετά από τον παραπάνω ορισμό των ενεργειών του κάθε χρήστη αντιλαμβανόμαστε ότι δημιουργούνται δύο νέες οντότητες στο σύστημα, το ραντεβού και το ωράριο εργασίας του ιατρού. Οι δύο αυτές οντότητες αποτελούν τους δύο επιπλέον μεσάζοντες πίνακες οι οποίοι θα κατασκευαστούν ως συνδετικός κρίκος των τριών χρηστών στην βάση δεδομένων.



Εικόνα 7 Πίνακες appointments & doctor_schedule της βάσης δεδομένων medical

Θα παρατηρήσουμε και από την παραπάνω εικόνα ότι ο πίνακας ραντεβού ενώνει τους δύο πίνακες του ιατρού και του ασθενή μέσω των πρωτευόντων κλειδιών τους επιτρέποντας έτσι την καταχώρηση των ραντεβού ως εγγραφές στον πίνακα με κοινά στοιχεία τα id του ιατρού και του ασθενή.

Ενώ στον πίνακα του ωραρίου εργασίας παρατηρείται ότι καλείται το πρωτεύων κλειδί του πίνακα ιατρού με σκοπό την καταχώρηση του ωραρίου σαν εγγραφή βάσει του id του κάθε ιατρού.

Συμπερασματικά η βάση δεδομένων του συστήματος αποτελείται από πέντε στο σύνολο πίνακες, οι τρείς εκ των οποίων είναι τα διαφορετικά είδη χρηστών και οι υπόλοιποι δυο πίνακες είναι οι συνδετικοί κρίκοι μεταξύ των χρηστών και των επιτρεπόμενων λειτουργιών τους.

7.2.1.2. Σύνδεση της βάσης δεδομένων με τα αρχεία

Μετά τον ορισμό και την κατασκευή των πινάκων στην βάση δεδομένων, σειρά έχει η σύνδεση της βάσης με τα αρχεία php του κώδικα μας.

Η σύνδεση επιτυγχάνεται μέσω του κεντρικού αρχείου appointment.php που βρίσκεται στον φάκελο classes. Το αρχείο αυτό αποτελεί την «καρδιά» της πλατφόρμας και αυτό γιατί μέσα εκεί συντάσσεται ο constructor της σύνδεσης με την βάση και ορίζονται οι βασικές μέθοδοι για την ανάκληση των δεδομένων.

7.2.2. Ανάλυση του τρόπου λειτουργίας των αρχείων

Σε αυτό το κεφάλαιο θα αναλύσουμε με λεπτομέρεια τον τρόπο με το οποίο λειτουργεί το κάθε αρχείο, πως αλληλοεπιδρούν μεταξύ τους και τελικά ποιο είναι το αποτέλεσμα που μας παρουσιάζουν στην οθόνη του υπολογιστή.

Εισάγοντας την διεύθυνση url: http://localhost/medical/index.php στον browser μεταβαίνουμε στην κεντρική σελίδα της πλατφόρμας που δεν είναι άλλο από το αρχείο index.php. Το αρχείο αυτό είναι η κεντρική σελίδα μέσα από την οποία μας δίνονται δύο επιλογές είτε να συνδεθούμε, είτε να προβούμε στην εγγραφή μας.

Επιλέγοντας τον σύνδεσμο εγγραφή οδηγούμαστε στην φόρμα εγγραφής ως ασθενής, η οποία κατασκευάζεται από το αρχείο signup.php. Συμπληρώνοντας τα στοιχεία μας και πατώντας το κουμπί εγγραφή καλείται το αρχείο signup.js μέσα από τον φάκελο js, όπου αναλαμβάνει να ενεργοποιήσει τον επιβεβαιωτή – parsley για την ορθή καταχώρηση των στοιχείων στην φόρμα και να καλέσει με την σειρά του το αρχείο signup_action.php που βρίσκεται στον υποφάκελο actions του φακέλου classes.

Κατά την διάρκεια της ανάλυσης θα παρατηρήσουμε ότι έχουμε κατασκευάσει τα αρχεία με τις τρεις διαφορετικές μορφές. Υπάρχουν τα αρχεία php που σκοπό έχουν να δημιουργήσουν το layout της κάθε σελίδας και να καλέσουν τα αντίστοιχα αρχεία javascript. Αυτά με την σειρά τους αναλαμβάνουν να εκτελέσουν τους επιβεβαιωτές και να καλέσουν την τρίτη μορφή αρχείων τα actions. Τα actions είναι αρχεία php στα οποία περιλαμβάνεται ο αντίστοιχος κώδικας για την backend επικοινωνία με τον server και την εκτέλεση των διάφορων queries στην βάση δεδομένων.

Έτσι έχουμε για παράδειγμα το αρχείο signup.php που δημιουργεί το layout της φόρμας εγγραφής και καλεί το αρχείο signup.js όπου επιβεβαιώνει ότι συμπληρώθηκαν ορθά όλα τα στοιχεία της φόρμας και έπειτα εκτελεί το αρχείο signup_action.php οπού αναλαμβάνει την καταχώρηση των στοιχείων της φόρμας στην βάση δεδομένων.

Αν επιλέξουμε την σύνδεση τότε καλείται το αρχείο login.php και προβάλλεται στην οθόνη μας η φόρμα σύνδεσης που μας ζητά μια ηλεκτρονική διεύθυνση και ένα κωδικό που έχουμε καταχωρήσει για τον λογαριασμό μας. Πατώντας το κουμπί σύνδεσης καλείται το αρχείο login.js ώστε να προβεί στην εκτέλεση του login_action.php. Το τελευταίο αρχείο αναλαμβάνει να ψάξει μέσα στις εγγραφές του πίνακα admin αν υπάρχει εκείνη η εγγραφή που αποτελείται από την ηλεκτρονική διεύθυνση και τον κωδικό που έχουμε εισάγει στην φόρμα. Αν δεν υπάρχει τότε αρχίζει την αναζήτηση στον πίνακα doctor. Αν δεν καταφέρει να βρει την εγγραφή και σε αυτό τον πίνακα τότε την αναζητά στον πίνακα patient. Στην περίπτωση που η εγγραφή δεν βρεθεί σε κανέναν πίνακα τότε εμφανίζεται μήνυμα στον χρήστη ότι έχει εισάγει λάθος στοιχεία σύνδεσης. Σε αντίθετη περίπτωση καλείται το αρχείο dashboard.php του αντίστοιχού χρήστη ανάλογα με το αν έχει συνδεθεί ως admin, doctor ή patient.

Τα dashboards έχουν διαφορετικό χρώμα μορφοποίησης για εύκολη και γρήγορη αντίληψη του χρήστη όσον αφορά τον ρόλο του στο σύστημα, δηλαδή αν έχει συνδεθεί ως admin τότε το dashboard θα είναι μπλε γαλάζιο, ως doctor πράσινο και ως patient τυρκουάζ.

7.2.2.1. Σύνδεση ως Διαχειριστής

Όπως αναφέραμε και παραπάνω αν ο χρήστης έχει καταφέρει και έχει εισάγει τα σωστά στοιχεία σύνδεσης τότε θα μεταφερθεί στο αντίστοιχο dashboard ανάλογα με τον ρόλο που έχει στο σύστημα. Αν λοιπόν έχει συνδεθεί ως διαχειριστής τότε θα δει στην οθόνη του το μπλε γαλάζιο dashboard με ένα κεντρικό μήνυμα καλωσορίσματος και το μενού του διαχειριστή στα αριστερά της οθόνης.

Το μενού αποτελείται από έξι κουμπιά όπου το καθένα οδηγεί στην αντίστοιχη σελίδα που αναφέρεται.

Το κουμπί «Αρχική» αν πατηθεί τότε καλείται εκ νέου το dashboard.php στον φάκελο admin.

Το κουμπί «Στατιστικά» μας οδηγεί στην σελίδα admin_appointment_stats.php, στην οποία προβάλλονται στατιστικά στοιχεία όπως ο αριθμός των νέων ραντεβού την τελευταία εβδομάδα, ο αριθμός των νέων εγγεγραμμένων ασθενών και των νέων καταχωρημένων ιατρών τις τελευταίες επτά ημέρες. Τα στατιστικά αυτά στοιχεία υπολογίζονται μέσω των συναρτήσεων που καλούνται από το αρχείο appointment.php.

Το κουμπί «Προφίλ» καλεί την σελίδα profile.php μέσω της οποίας κατασκευάζεται μια φόρμα με τα στοιχεία του admin που είναι καταχωρημένα μέσα στην βάση. Στο επάνω μέρος παρατηρούμε την ύπαρξη ενός κουμπιού που μας προτρέπει στην μεταβολή των στοιχείων. Αν το πατήσουμε τότε καλείται το αρχείο edit_admin_profile.js το οποίο με την σειρά του επιβεβαιώνει ότι στην φόρμα έχουν συμπληρωθεί όλα τα πεδία ορθά και καλεί το αρχείο edit_admin_profile_action.php. Το συγκεκριμένο αρχείο αναλαμβάνει να εκτελέσει το query για την ανανέωση του πίνακα στην βάση δεδομένων όταν πατηθεί το κουμπί submit. Αν τα στοιχεία έχουν συμπληρωθεί σωστά τότε θα εμφανιστεί μήνυμα επιβεβαίωσης στον χρήστη διαφορετικά το σύστημα θα τον ενημερώσει την τυχόν ύπαρξη λάθους.

Το κουμπί «Γιατροί» αναλαμβάνει να μας οδηγήσει στην σελίδα admin_doctors.php. Στην συγκεκριμένη σελίδα παρουσιάζεται ένας πίνακας με τους καταχωρημένους γιατρούς στην βάση δεδομένων. Για να κατασκευαστεί ο πίνακας αυτός και να παρουσιάσει τις κατάλληλες εγγραφές καλείται το αρχείο admin_dash_doctors_table.js. Στο αρχείο αυτό ορίζεται ο πίνακας ως datatable και καθορίζονται οι διαδικασίες που θα ακολουθηθούν σε περίπτωση που επιλεγεί κάποιο κουμπί από τα τρία που υπάρχουν στο δεξί μέρος του πίνακα. Κάθε διαδικασία καλεί την αντίστοιχη συνάρτηση από το αρχείο admin_dash_doctors_table_action.php.

Τα τρία κουμπιά που υπάρχουν στο δεξί μέρος του πίνακα χρησιμοποιούνται για την παρουσίαση των στοιχείων του ιατρού, την διαγραφή του καθώς και την αλλαγή των στοιχείων του προφίλ του.

Αν πατήσουμε το μπλε κουμπί της παρουσίασης των στοιχείων θα εμφανιστεί ένα modal με τα προσωπικά στοιχεία του ιατρού.

Στην περίπτωση που πατήσουμε το κίτρινο κουμπί της μεταβολής τότε θα εμφανιστεί σε modal μια φόρμα όπου θα έχουμε την δυνατότητα να αλλάξουμε τα στοιχεία της εγγραφής ενώ σε περίπτωση που επιλέξουμε το κόκκινο κουμπί της ακύρωσης θα παρουσιαστεί μήνυμα προτροπής στον χρήστη ρωτώντας τον αν θέλει να διαγράψει τον συγκεκριμένο ιατρό.

Στο επάνω μέρος από τον πίνακα παρατηρείτε ότι υπάρχει το πράσινο κουμπί της καταχώρησης νέου ιατρού. Αν το επιλέξουμε τότε θα εμφανιστεί σε modal μια φόρμα εισαγωγής στοιχείων.

Το κουμπί «Ασθενείς» μας οδηγεί στην σελίδα admin_patients.php στην οποία προβάλλεται ένας πίνακας με όλες τις εγγραφές των ασθενών. Για να κατασκευαστεί ο πίνακας καλείται το αρχείο admin_dash_patients_table.js το οποίο ορίζει την δημιουργία του πίνακα ως αντικείμενο datatable και καλεί για την συμπλήρωση του το αρχείο admin_dash_patients_table_action.php. Το τελευταίο

είναι υπεύθυνο για την εκτέλεση του query που αφορά την ανάκληση των δεδομένων από το πίνακα patient της βάσης δεδομένων.

Τέλος το κουμπί «Λίστα Ραντεβού» χρησιμοποιείται για να μεταβούμε στην ιστοσελίδα admin_appointment_list.php μέσω της οποίας παρουσιάζεται ένας πίνακας με τα ραντεβού που έχουν καταχωρηθεί στην βάση. Όπως παρατηρούμε στα δεξιά του πίνακα υπάρχει η σήμανση για την κατάσταση του ραντεβού καθώς και ένα κουμπί παρουσίασης των στοιχείων του συγκεκριμένου ραντεβού. Οι συγκεκριμένες λειτουργίες καθορίζονται μέσα από το αρχείο admin_ appointment_list.js ενώ το query ανάκλησης των δεδομένων εκτελείται από το admin_appointment_list_action.php.

Αξίζει να σημειωθεί ότι στο επάνω μέρος από τον πίνακα υπάρχει αναζήτηση βάσει της ημερομηνίας καταχώρησης του ραντεβού. Αρκεί να εισάγει ο χρήστης τις ημερομηνίες από και μέχρι και να πατήσει το κουμπί της αναζήτησης ώστε να δει την λίστα με τα ραντεβού μέσα στο χρονικό διάστημα που έχει επιλέξει.

7.2.2.2. Σύνδεση ως Ιατρός

Στην περίπτωση που ο χρήστης συνδεθεί ως ιατρός θα δει το dashboard σε χρώμα πράσινο. Στην Αρχική σελίδα για ακόμη μια φορά υπάρχει το μήνυμα καλωσορίσματος αναφέροντας το όνομα του χρήστη, ενώ στα αριστερά το μενού περιλαμβάνει τρία κουμπιά αυτήν την φορά.

Το κουμπί «Αρχική» όπως αναφέραμε και πιο πάνω μας οδηγεί και πάλι στο dashboard του χρήστη, ενώ το κουμπί «Προφίλ» μας παρουσιάζει την φόρμα με τα στοιχεία του ιατρού η οποία ενεργοποιείται αν πατήσουμε το κουμπί της αλλαγής στοιχείων. Η σελίδα profile.php του ιατρού καλεί το αρχείο edit_doctor_profile.js για την επιβεβαίωση της ορθής συμπλήρωσης των πεδίων της φόρμας καθώς και για την εκτέλεση των queries μέσω του αρχείου edit_doctor_profile_action.php.

Επιπρόσθετα το κουμπί «Ωράριο Εργασίας» μας παρουσιάζει την σελίδα doctor_working_hours_list.php στην οποία εμφανίζεται ένας πίνακας με τις εγγραφές του ωραρίου εργασίας του συγκεκριμένου ιατρού. Παρέχεται η δυνατότητα να αλλάξουμε την κατάσταση του ωραρίου από ενεργό σε ανενεργό και το αντίστροφο, να μεταβάλλουμε την συγκεκριμένη εγγραφή καθώς και να την διαγράψουμε πλήρως μέσω των κουμπιών που υπάρχουν στην τελευταία στήλη του πίνακα. Πατώντας το πράσινο κουμπί στο επάνω μέρος της σελίδας μπορούμε να καταχωρήσουμε ένα νέο ωράριο στην φόρμα που εμφανίζεται με την μορφή modal.

Όλες αυτές οι ενέργειες που αναφέρθηκαν ορίζονται στα αρχεία doctor_working_hours_list.js και doctor_working_hours_list_action.php.

7.2.2.3. Σύνδεση ως Ασθενής

Στην τρίτη και τελευταία περίπτωση που ο χρήστης συνδεθεί ως ασθενής παρατηρούμε ότι το dashboard είναι σε χρώμα τυρκουάζ. Επαναλαμβάνουμε ότι αναφέρεται το χρώμα του dashboard γιατί βοηθά να αντιληφθούμε με ποιο ρόλο έχουμε συνδεθεί στην πλατφόρμα.

Το μενού επιλογών του ασθενή περιλαμβάνει τέσσερα κουμπιά. Το πρώτο είναι το κλασικό κουμπί της αρχικής σελίδας ενώ το δεύτερο του προφίλ. Όσον αφορά το προφίλ του ασθενή καλούνται τα αρχεία edit_patient_profile.js και edit_patient_profile_action.php.

Το κουμπί «Ωράριο Γιατρών» μας οδηγοί στην σελίδα patient_book_appointment.php μέσα στην οποία εμφανίζεται ο πίνακας με τα διαθέσιμα καταχωρημένα ωράρια ιατρών. Δίνεται η δυνατότητα να κλείσει ένα ραντεβού ο ασθενής σε κάποιον συγκεκριμένο γιατρό πατώντας το κουμπί «Καταχώρηση Ραντεβού». Με το πάτημα του κουμπιού καλείται το αρχείο patient_book_appointment.js που αναλαμβάνει να εμφανίσει το modal με την φόρμα του ραντεβού. Μέσα στο modal υπάρχει πεδίο που αναγράφεται ο λόγος της επίσκεψης καθώς και το κουμπί της καταχώρησης. Αν επιλεγεί καταχώρηση τότε καλείται το αρχείο patient_book_appointment_action.php για την εκτέλεση του query που θα εισάγει την εγγραφή στον πίνακα appointment της βάσης δεδομένων.

Τέλος το κουμπί «Τα Ραντεβού μου» μας εμφανίζει την σελίδα patient_appointment_list.php. Η συγκεκριμένη σελίδα αποτελείται από τον πίνακα των ραντεβού. Εδώ προβάλλονται όλα τα ραντεβού που έχει κλείσει ο χρήστης παρέχοντας την δυνατότητα να δει σε τι κατάσταση βρίσκονται, να «κατεβάσει» σε pdf τα στοιχεία του ραντεβού καθώς και να ακυρώσει κάποιο. Όλες οι διεργασίες ορίζονται στο αρχείο patient_appointment_list.js και τα ερωτήματα εκτελούνται από το αρχείο patient_appointment_list_action.php.

Οφείλουμε να τονίσουμε ότι κάθε χρήστης οποιουδήποτε ρόλου μπορεί να κάνει αποσύνδεση πατώντας το μενού που βρίσκεται στην επάνω δεξιά γωνία του dashboard. Το μενού αναδιπλώνεται και περιέχει δύο επιλογές, είτε να οδηγηθεί ο χρήστης στο προφίλ του είτε να κάνει οριστική αποσύνδεση από την πλατφόρμα. Η λειτουργία της αποσύνδεσης πραγματοποιείται από το αρχείο logout.php.

8. ΣΥΜΠΕΡΑΣΜΑΤΑ

Έχοντας αναλύσει και παρουσιάσει πλέον όλες τις γλώσσες προγραμματισμού που χρησιμοποιήσαμε για την κατασκευή της πλατφόρμας μας αλλά και τον τρόπο δημιουργίας της ίδιας της ιστοσελίδας έχουμε καταλήξει πλέον σε ορισμένα συμπεράσματα.

Η χρήση της javascript και ιδιαίτερα της απλουστευμένης μορφής της JQuery βοήθησε στο να έχουμε τον απόλυτο έλεγχο των κλήσεων των ποικίλων αρχείων php. Ο καταμερισμός των διεργασιών σε διαφορετικά αρχεία php παρέχει βελτίωση στην λειτουργικότητα της πλατφόρμας χαρίζοντας ευελιξία στην συγγραφή του κώδικα και στις διάφορες αλλαγές που απαιτούνται κάθε τόσο.

Σημαντική προσθήκη στην επιβεβαίωση των ηλεκτρονικών διευθύνσεων κατά την εγγραφή νέων χρηστών ήταν ο phpmailer ο οποίος εκτός από την γρήγορη εγκατάσταση και εύκολη χρήση του μας πρόσφερε μια υποτυπώδη εικόνα επιβεβαίωσης σε τοπικό επίπεδο χρησιμοποιώντας ένα απλό λογαριασμό email. Με την βοήθεια του mailer μπορέσαμε να δούμε σε εμβρυακό στάδιο πως δημιουργείς ένα απλό επιβεβαιωτικό μήνυμα και πως να δημιουργείς ένα κωδικό ενεργοποίησης, να τον αποθηκεύεις και να τον αποστέλλεις στο email που καταχωρεί ο χρήστης στην πλατφόρμα.

Ένα μειονέκτημα που εντοπίσαμε με την χρήση του phpmailer είναι ότι αν ο χρήστης χρησιμοποιήσει για πρόσβαση τον browser Chrome τότε απαιτείται να έχει ενεργοποιήσει στο προφίλ του την επιλογή «Allow less secure apps» που επιτρέπει στο script του mailer να εκτελεστεί κατασκευάζοντας τον κωδικό ενεργοποίησης που θα σταλεί στο email του χρήστη. Αυτή η απαίτηση αποτελεί μειονέκτημα για την χρήση του phpmailer καθώς ο κατασκευαστής της πλατφόρμας δεν είναι δυνατόν να γνωρίζει ποιον browser χρησιμοποιεί ο χρήστης που προσπαθεί να κάνει εγγραφή και αν ακόμη έχει ενεργοποιήσει την συγκεκριμένη επιλογή στο προφίλ του, με αποτέλεσμα ορισμένοι χρήστες να προσπαθήσουν να εγγραφούν αλλά να μην τους σταλεί ποτέ το μήνυμα της ενεργοποίησης του email. Φυσικά αυτό δεν θα εμποδίσει την σύνδεση στο σύστημα γιατί επιλέξαμε να συντάξουμε έτσι τον κώδικα ώστε να μην επηρεάζει την απευθείας σύνδεση.

Επιπλέον αξίζει να σημειωθεί ότι προσπαθήσαμε να δημιουργήσουμε μια κοινή φόρμα σύνδεσης και για τους τρείς ρόλους χρήστη (διαχειριστής, ιατρός, ασθενής) και αυτό γιατί μειώνει κατά πολύ τον κώδικα και τον όγκο των αρχείων όπως επίσης και αισθητικά παρέχει ευκολότερη και πιο κατανοητή πρόσβαση στο σύστημα από την μεριά του χρήστη. Ίσως το να χρησιμοποιήσουμε διαφορετική σελίδα για την πρόσβαση του διαχειριστή να μας έδινε καλύτερη κατανομή των

ρόλων και μεγαλύτερη αυτονομία στην βάση δεδομένων, προτιμήσαμε όμως στο σύστημα μας να κατασκευάσουμε μια και μοναδική φόρμα σύνδεσης πρώτον γιατί ο όγκος των δεδομένων της βάσης μας δεν πρόκειται να μεγαλώσει στο μέλλον καθότι χρησιμοποιείται μόνο για ακαδημαϊκούς λόγους, αλλά και δεύτερον γιατί θεωρήσαμε ότι αξίζει τον κόπο να προσπαθήσουμε να δημιουργήσουμε κάτι διαφορετικό καθώς η πλειοψηφία της βιβλιογραφίας στο διαδίκτυο παρουσιάζει δύο διαφορετικές φόρμες σύνδεσης. Η πρώτη φόρμα είναι για τον διαχειριστή και η δεύτερη για τον ασθενή και τον γιατρό.

Τελειώνοντας θεωρούμε ότι με την παρούσα εργασία διευρύναμε τον τρόπο κατασκευής και εξέλιξης ενός τέτοιου συστήματος ενώ ελπίζουμε στο μέλλον οι γνώσεις που αποκομίσαμε να μας επιτρέψουν την δημιουργία παρόμοιας πλατφόρμας στο επίπεδο του διαδικτύου και όχι στα στενά όρια του τοπικού δικτύου – localhost.

ΒΙΒΛΙΟΓΡΑΦΙΑ

Ιστοσελίδες

1. https://el.wikipedia.org/wiki/SQL
2. http://www.sqlcourse.com/intro.html
3. https://www.tutorialspoint.com/difference-between-delete-and-drop-sql
4. https://www.w3schools.com/sql/default.asp
5. https://el.wikipedia.org/wiki/Bootstrap
6. https://en.wikipedia.org/wiki/Usage_share_of_web_browsers
7. https://eclass.aegean.gr/modules/document/file.php
8. https://www.codeleaks.io/link-external-php-file-to-html/#
9. https://www.tutorialrepublic.com/php-tutorial/php-include-files.php
10. https://www.geeksforgeeks.org/what-is-the-difference-between-scss-and-sass/
11. https://www.csc.com.gr/sql-joins/

Πτυχιακές Εργασίες

1. Οικονόμου Αλέξιος (2017), Διπλωματική Εργασία «Τεχνολογίες Κατασκευής Διαδραστικών Σελίδων», Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης
2. Χατζηζαχαρίας Μηνάς (2019), Πτυχιακή Εργασία «Σχεδιασμός και υλοποίηση ιστοτόπου προβολής τουριστικών καταλυμάτων με χρήση προχωρημένων τεχνικών διαδικτυακού προγραμματισμού», Τεχνολογικό Εκπαιδευτικό Ίδρυμα Δυτικής Ελλάδας
3. Παπαδοπούλου Μαρία (2018), Πτυχιακή Εργασία «Σχεδιασμός και ανάπτυξη δικτυακής εφαρμογής ηλεκτρονικής παρακολούθησης κατάστασης αντικειμένων προς επισκευή», Πανεπιστήμιο Πελοποννήσου
4. Ανθιμίδης Νίκος, Κουπτσίδης Αβρααμ, Πτυχιακή Εργασία «Κατασκευή Website Παρουσίασης Κατηγοριοποιημένων Αντικειμένων Με Custom Web Service», Αλεξάνδρειο Τ.Ε.Ι. Θεσσαλονίκης

5. Γεργατσούλης Μανόλης, Παρουσίαση «Το σχεσιακό μοντέλο και η γλώσσα SQL», Ιόνιο Πανεπιστήμιο
6. Γεωργιάδης Παναγιώτης (2016), Πτυχιακή Εργασία «Σχεδιασμός και Δημιουργία Εφαρμογής για το Τεχνολογικό Πανεπιστήμιο Κύπρου», Τεχνολογικό Πανεπιστήμιο Κύπρου
7. Μαρκόπουλος Δημήτριος (2006), Πτυχιακή Εργασία «Εφαρμογή διαδικτυακού επεξεργαστή κειμένου & html (web-editor) στη γλώσσα προγραμματισμού JavaScript», Τεχνολογικό Εκπαιδευτικό Ίδρυμα Καβάλας

Documentation Γλωσσών Προγραμματισμού

1. https://www.php.net/
2. https://sass-lang.com/guide
3. https://www.javascript.com/