ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΑΝΑΠΤΥΞΗ ΔΙΑΔΙΚΤΥΑΚΗΣ ΠΛΑΤΦΟΡΜΑΣ ΓΙΑ ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ ΣΤΗΝ ΔΕΥΤΕΡΟΒΑΘΜΙΑ ΕΚΠΑΙΔΕΥΣΗ

ΤΟΥΜΠΑ ΓΕΩΡΓΙΑ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΥΠΕΥΘΥΝΟΣ

ΤΖΙΑΛΛΑΣ ΓΡΗΓΟΡΙΟΣ
.....Καθηγητής.....

Λαμία ..... έτος 2024

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΑΝΑΠΤΥΞΗ ΔΙΑΔΙΚΤΥΑΚΗΣ ΠΛΑΤΦΟΡΜΑΣ ΓΙΑ ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ ΣΤΗΝ ΔΕΥΤΕΡΟΒΑΘΜΙΑ ΕΚΠΑΙΔΕΥΣΗ

ΤΟΥΜΠΑ ΓΕΩΡΓΙΑ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΥΠΕΥΘΥΝΟΣ

ΤΖΙΑΛΛΑΣ ΓΡΗΓΟΡΙΟΣ
.....Καθηγητής.....

Λαμία ..... έτος 2024

UNIVERSITY OF
THESSALY

SCHOOL OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE & TELECOMMUNICATIONS

DEVELOPMENT OF AN ONLINE PLATFORM FOR INTRODUCTION TO SECONDARY EDUCATION PROGRAMMING

TOUMPA GEORGIA

FINAL THESIS

ADVISOR

TZIALLAS GRIGORIOS
.....Professor.....

Lamia ..... year 2024

«Με ατομική μου ευθύνη και γνωρίζοντας τις κυρώσεις [(1)], που προβλέπονται από της διατάξεις της παρ. 6 του άρθρου 22 του Ν. 1599/1986, δηλώνω ότι:

*1. Δεν παραθέτω κομμάτια βιβλίων ή άρθρων ή εργασιών άλλων αυτολεξεί **χωρίς να τα περικλείω σε εισαγωγικά** και χωρίς να αναφέρω το συγγραφέα, τη χρονολογία, τη σελίδα. Η αυτολεξεί παράθεση χωρίς εισαγωγικά χωρίς αναφορά στην πηγή, είναι λογοκλοπή. Πέραν της αυτολεξεί παράθεσης, λογοκλοπή θεωρείται και η παράφραση εδαφίων από έργα άλλων, συμπεριλαμβανομένων και έργων συμφοιτητών μου, καθώς και η παράθεση στοιχείων που άλλοι συνέλεξαν ή επεξεργάσθηκαν, χωρίς αναφορά στην πηγή. Αναφέρω πάντοτε με πληρότητα την πηγή κάτω από τον πίνακα ή σχέδιο, όπως στα παραθέματα.*

*2. Δέχομαι ότι η αυτολεξεί **παράθεση χωρίς εισαγωγικά**, ακόμα κι αν συνοδεύεται από αναφορά στην πηγή σε κάποιο άλλο σημείο του κειμένου ή στο τέλος του, είναι αντιγραφή. Η αναφορά στην πηγή στο τέλος π.χ. μιας παραγράφου ή μιας σελίδας, δεν δικαιολογεί συρραφή εδαφίων έργου άλλου συγγραφέα, έστω και παραφρασμένων, και παρουσίασή τους ως δική μου εργασία.*

3. Δέχομαι ότι υπάρχει επίσης περιορισμός στο μέγεθος και στη συχνότητα των παραθεμάτων που μπορώ να εντάξω στην εργασία μου εντός εισαγωγικών. Κάθε μεγάλο παράθεμα (π.χ. σε πίνακα ή πλαίσιο, κλπ.), προϋποθέτει ειδικές ρυθμίσεις, και όταν δημοσιεύεται προϋποθέτει την άδεια του συγγραφέα ή του εκδότη. Το ίδιο και οι πίνακες και τα σχέδια

4. Δέχομαι όλες τις συνέπειες σε περίπτωση λογοκλοπής ή αντιγραφής.

Ημερομηνία: 21/10/2024

– Η Δηλ.

(1) «Όποιος εν γνώσει του δηλώνει ψευδή γεγονότα ή αρνείται ή αποκρύπτει τα αληθινά με έγγραφη υπεύθυνη δήλωση
του άρθρου 8 παρ. 4 Ν. 1599/1986 τιμωρείται με φυλάκιση τουλάχιστον τριών μηνών. Εάν ο υπαίτιος αυτών των πράξεων
σκόπευε να προσπορίσει στον εαυτόν του ή σε άλλον περιουσιακό όφελος βλάπτοντας τρίτον ή σκόπευε να βλάψει άλλον, τιμωρείται με κάθειρξη μέχρι 10 ετών.»

ΠΕΡΙΛΗΨΗ

Στην παρούσα πτυχιακή εργασία παρουσιάζεται η διαδικασία ανάπτυξης μίας διαδικτυακής πλατφόρμας εκμάθησης της προγραμματιστικής γλώσσας C. Η πλατφόρμα αυτή επικεντρώνεται στην δημιουργία μια ολοκληρωμένης σειράς μαθημάτων, τα οποία είναι χωρισμένα σε δώδεκα ενότητες. Είναι σχεδιασμένη για αρχάριους προγραμματιστές και κυρίως για παιδιά της Δευτεροβάθμιας Εκπαίδευσης που επιθυμούν να μάθουν τις βασικές αρχές του προγραμματισμού. Η ιστοσελίδα προσφέρει ένα εύχρηστο περιβάλλον εκμάθησης, στο οποίο οι χρήστες πρέπει να συνδεθούν για να αποκτήσουν πρόσβαση σε ποικιλία εκπαιδευτικού υλικού. Να σημειωθεί ότι η γλώσσα C περιέχει μεγάλο εύρος δυνατοτήτων το οποίο στα πλαίσια της πτυχιακής εργασίας δεν έχει υλοποιηθεί ολόκληρο, με σκοπό η διαδικτυακή εφαρμογή να είναι πιο απλή και κατανοητή στο κοινό που απευθύνεται. Στην εργασία αυτή παρουσιάζεται ο τρόπος υλοποίησης της ιστοσελίδας με κύρια μέσα προγραμματισμού Html/Css/JavaScript, Node.js και μιας βάσης δεδομένων. Στόχος της εφαρμογής είναι να εμπνεύσει νέους και νέες που θέλουν να έχουν μια πρώτη επαφή με τον προγραμματισμό, δίνοντας τους την δυνατότητα να προσαρμόζουν τον ρυθμό της μελέτης τους και να αξιολογούν οι ίδιοι -μέσω τεστ- την κατανόηση τους στην θεωρητική παράδοση των εννοιών της γλώσσας C.

ABSTRACT

This thesis presents the process of developing an online learning platform for C programming language. This platform focuses on the creation of a completed series of courses, which are divided into twelve sections. The website is designed for beginner programmers and especially for Secondary School students who wish to learn the basics of programming. It offers an easy-to-use learning environment where users must log in to access a variety of educational materials. It should be noted that C language contains a wide range of possibilities, which in the context of the thesis has not been fully implemented, in order to make the application simpler and more understandable to the targeted audience. In the thesis is presented the way that the website was created with the main programming tools being: Html/CSS/JavaScript, Node.js and a database. The purpose of the application is to inspire young men and women who want to learn programming for the first time while giving them the possibility to adjust the pace of their study and to evaluate themselves -through tests- their understanding in the theoretical tradition of the concepts of C language.

Table of Contents

ΚΕΦΑΛΑΙΟ 1 Εισαγωγή

1.1: ΕΙΣΑΓΩΓΗ

Θέμα της πτυχιακής εργασίας αυτής είναι: η ανάπτυξη μιας διαδραστικής πλατφόρμας για αρχάριους προγραμματιστές με σκοπό την εκμάθηση μιας πολύ δημοφιλής γλώσσας προγραμματισμού, την γλώσσα C. Υπάρχουν πολλές διαθέσιμες πηγές μάθησης προγραμματισμού από online εργαλεία μέχρι δια ζώσεις μάθηση, στα οποία μπορεί ο καθένας να κάνει την δικιά του έρευνα και να μάθει κάποια γλώσσα προγραμματισμού. Στα πλαίσια της πτυχιακής εργασία αναπτύχθηκε ένα ακόμη μέσο εκμάθησης προγραμματισμού και συγκεκριμένα για την γλώσσα προγραμματισμού C, το οποίο αποτελείται από μια ιστοσελίδα. Για την ανάπτυξη της πλατφόρμας αυτής χρειάστηκε να συνδυαστούν πολλές τεχνολογίες ανάπτυξης εφαρμογών για την δημιουργία του front end μέρους της εφαρμογής, καθώς και για το back end μέρος της. Κάποια μέσα από αυτά μπορεί να είναι: Html, JavaScript, express, Json, php, node.js, AJAX και πολλά άλλα ακόμη.

Η ιστοσελίδα περιέχει εκπαιδευτικό υλικό, όπως βίντεο, άρθρα και το κύριο μέρος της είναι η παράθεση θεωρητικού μέρους της γλώσσας C χωρισμένο σε δώδεκα διαφορετικές ενότητες- όπου ένας χρήστης μπορεί να έχει πρόσβαση σε όλες αυτές ανεξαρτήτως σειράς για δικιά του ευκολία. Περιέχει επίσης ένα μεγάλο μέρος εξέτασης του θεωρητικού περιεχομένου, σε μορφή τεστ και κουίζ κλειστού και ανοιχτού τύπου για καλύτερη κατανόηση των όσων μαθαίνει. Για να έχει ένας χρήστης πρόσβαση στην πλατφόρμα θα χρειαστεί να κάνει εγγραφή και έπειτα σύνδεση με τα στοιχεία του, ώστε να έχει πρόσβαση στο εκπαιδευτικό υλικό. Η ιστοσελίδα προσαρμόζεται στο προφίλ του κάθε χρήστη, εμφανίζοντας του την επίδοση του κατά την διάρκεια των μαθημάτων. Σε κάθε ενότητα υπάρχουν παραδείγματα κώδικα, τα οποία μπορεί ο χρήστης να δοκιμάσει να υλοποιήσει και ο ίδιος όχι μέσα από την ιστοσελίδα αλλά από κάποιο εργαλείο προγραμματισμού. Επιπλέον, δίνεται η δυνατότητα ανάπτυξης μεγάλων προγραμμάτων σε μερικά από τα κεφάλαια, καθώς και ένα τελικό project που περιέχει κομμάτια από όλες τις ενότητες. Τέλος, να σημειωθεί ότι η γλώσσα C περιέχει μεγάλο εύρος δυνατοτήτων το οποίο στα πλαίσια της πτυχιακής εργασίας δεν έχει υλοποιηθεί ολόκληρο, με σκοπό η διαδικτυακή εφαρμογή να είναι πιο απλή και κατανοητή στο κοινό που απευθύνεται.

Στην παρούσα πτυχιακή εργασία, εστιάζουμε στην ανάπτυξη μιας διαδικτυακής πλατφόρμας για την εκμάθηση προγραμματισμού, αξιοποιώντας κάποιες από τις σύγχρονες τεχνολογίες που αναφέρθηκαν προηγουμένως. Η εφαρμογή βασίζεται στη χρήση δύο διακομιστών (servers) του Node.js για την διαχείριση της λογικής του συστήματος της ιστοσελίδας και του PostgreSQL για την αποθήκευση πληροφοριών και δεδομένων του χρήστη για μελλοντική επεξεργασία και χρήση τους. Οι δύο αυτοί servers λειτουργούν παράλληλα σε τοπικό περιβάλλον (localhost), εξασφαλίζοντας την επικοινωνία και διαχείριση των δεδομένων μέσω μίας πλατφόρμας διαχείρισης βάσεων δεδομένων. Για τον σκοπό αυτό επιλέχθηκε η pgAdmin4.

1.2: ΣΤΟΧΟΣ

Η εργασία αυτή στοχεύει να καλύψει τόσο τις θεωρητικές, όσο και τις πρακτικές πτυχές ανάπτυξης εφαρμογών εκμάθησης προγραμματισμού και συγκεκριμένα μια διαδικτυακή πλατφόρμα, παρέχοντας μια ολοκληρωμένη προσέγγιση στην κατασκευή μιας αποδοτικής εκπαιδευτικής εφαρμογής.

Στόχος της εφαρμογής είναι να υποστηρίξει την αυτόνομη μάθηση μαθητών και αρχάριων προγραμματιστών, δίνοντας τους την δυνατότητα να προσαρμόσουν τον ρυθμό της μελέτης τους, να αξιολογούνται και να κατανοήσουν τις βασικές αρχές του προγραμματισμού στην γλώσσα C.

ΚΕΦΑΛΑΙΟ 2 ΕΦΑΡΜΟΓΕΣ ΕΚΜΑΘΗΣΗΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

2.1: ΒΑΣΙΚΑ ΣΤΟΙΧΕΙΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

Τι είναι ένα **πρόβλημα**;

Με τον όρο πρόβλημα προσδιορίζεται μια κατάσταση η οποία χρήζει αντιμετώπισης. Ένα πρόβλημα σχεδόν πάντα ακολουθείται από μια λύση. Η λύση αυτή άλλες φορές είναι προφανής και άλλες φορές πρέπει να ψάξουμε να την βρούμε. Στην καθημερινότητα μας ερχόμαστε όλοι αντιμέτωποι με προβλήματα, μικρά και μεγάλα, τα οποία ψάχνουμε να επιλύσουμε. Ο υπολογιστής έχει και αυτός τα δικά του προβλήματα, τα λεγόμενα υπολογιστικά προβλήματα.

Η επίλυση ενός προβλήματος με τον υπολογιστή περιλαμβάνει: τον προσδιορισμό του προβλήματος, την ανάπτυξη της λύσης του και τέλος την μετατροπή της λύσης αυτής σε μια μορφή κατανοητή από τον υπολογιστή. Ο υπολογιστής είναι ένα μηχάνημα, το οποίο δεν μπορεί να καταλάβει ενέργειες, λέξεις και εντολές όπως τις καταλαβαίνει ένας άνθρωπος. Οι σύγχρονοι υπολογιστές αντιλαμβάνονται μόνο το δυαδικό σύστημα (binary system), που αποτελείται από τα ψηφία 0 και 1. Όλες, λοιπόν, οι εντολές που δίνουμε σε έναν υπολογιστή αποθηκεύονται ως μια σειρά αριθμών σε αναπαράσταση των 0 και 1, την ευρέως γνωστή γλώσσα μηχανής.

Τι είναι **αλγόριθμος**;

Αλγόριθμο ονομάζουμε μια πεπερασμένη, σαφή και ακριβή περιγραφή μιας σειράς ενεργειών, που είναι αυστηρά καθορισμένες, με σκοπό την επίλυση ενός προβλήματος. Όταν σχεδιάζουμε έναν αλγόριθμο, πρέπει να είμαστε ιδιαίτερα προσεκτικοί, να βάζουμε με λογική σειρά τις οδηγίες αυτές που θα μας οδηγήσουν στη λύση του προβλήματός μας. Αφού γίνει η ανάπτυξη του αλγορίθμου, δηλαδή η επίλυση ενός προβλήματος σε μορφή εντολών, σειρά έχει η μετατροπή του σε μορφή κατανοητή από τον υπολογιστή και πιο συγκεκριμένα, η διατύπωση των εντολών του αλγορίθμου στο δυαδικό σύστημα. Το εργαλείο μετατροπής αυτό λέγεται μεταγλωττιστής.

Ο **προγραμματισμός**, ασχολείται με την δημιουργία του προγράμματος, το σύνολο δηλαδή εντολών το οποίο γράφει ο προγραμματιστής με στόχο την επίλυση του προβλήματος και που έπειτα πρόκειται να δοθεί στον υπολογιστή για να γίνει η μεταγλώττιση. Ένα **πρόγραμμα** λοιπόν, είναι η αναπαράσταση ενός αλγορίθμου γραμμένη σε γλώσσα κατανοητή για έναν υπολογιστή. Αποτελείται από μία σειρά εντολών που δίνονται στον υπολογιστή με σκοπό αυτός να εκτελέσει κάποια συγκεκριμένη λειτουργία ή να υπολογίσει κάποιο επιθυμητό αποτέλεσμα. Η εργασία σύνταξης των προγραμμάτων ονομάζεται προγραμματισμός. Κάθε πρόγραμμα γράφεται σε κάποια **γλώσσα προγραμματισμού**. Κάθε γλώσσα προγραμματισμού διαφέρει σε συγκεκριμένα στοιχεία, όπως στην σύνταξη, στο λεξιλόγιο, στην δομή κ.α. Ένα πρόβλημα όπως γνωρίζουμε ήδη και από την καθημερινότητα μας δεν έχει πολλές φορές μόνο μία λύση. Έτσι και με τα προβλήματα με υπολογιστή, έχουν διάφορες λύσεις είτε με την ίδια, είτε και με διαφορετική γλώσσα προγραμματισμού. Σχεδόν όλες οι γλώσσες προγραμματισμού μοιράζονται πολλά κοινά

χαρακτηριστικά. Παρόμοιες εντολές, δομή προγράμματος, τύποι δεδομένων και άλλα είναι μερικά τέτοια παραδείγματα. Τα προγράμματα που δημιουργεί ένας προγραμματιστής πρέπει να τηρούν βασικούς γραμματικούς και συντακτικούς κανόνες. Οι κανόνες πρέπει να τηρούνται αυστηρά και στην συνέχεια ελέγχονται από τον μεταγλωττιστή ή τον διερμηνευτή (compiler) για τυχόν λάθη, τα οποία έχουν να κάνουν κυρίως με την σύνταξη του προγράμματος και όχι μόνο.

- Ο μεταγλωττιστής είναι το λογισμικό που επιτυγχάνει την μετάφραση ενός προγράμματος από μια γλώσσα σε μία άλλη. Χρησιμοποιείται για την μετατροπή ενός προγράμματος γραμμένου σε **γλώσσα προγραμματισμού υψηλού επιπέδου**, σε εντολές γλώσσας μηχανής. Ως είσοδο δέχεται ένα πρόγραμμα που ονομάζεται **πηγαίο πρόγραμμα** και παράγει ένα ισοδύναμο σε γλώσσα μηχανής, το **αντικείμενο πρόγραμμα**.

- Ο διερμηνευτής από την άλλη, είναι και αυτό ένα παρόμοιο μεταφραστικό πρόγραμμα που χρησιμοποιείται για την μετατροπή ενός προγράμματος μιας υψηλού επιπέδου γλώσσας προγραμματισμού, σε εντολές γλώσσας μηχανής. Διαβάζει μια προς μια τις εντολές του πηγαίου προγράμματος και για κάθε εντολή εκτελεί αμέσως την ισοδύναμη ακολουθία εντολών του αντικείμενου προγράμματος.

Η διαδικασία αυτή της μεταγλώττισης δεν είναι μια απλή υπόθεση. Με απλά λόγια ο προγραμματιστής φτιάχνει τον κώδικα, τις εντολές για το πηγαίο πρόγραμμα σε κάποια γλώσσα προγραμματισμού. Το πηγαίο πρόγραμμα τοποθετείται ως είσοδος στον μεταγλωττιστή για να εντοπιστούν -αν υπάρχουν- συντακτικά λάθη και το εργαλείο αυτό εμφανίζει κατάλληλα μηνύματα για αυτά. Το κάθε λάθος απαιτεί διόρθωση από τον προγραμματιστή και η διαδικασία επαναλαμβάνεται έως ότου δεν υπάρχουν πια λάθη στον πηγαίο κώδικα. Το πρόγραμμα που παράγει ο μεταγλωττιστής αναφέρθηκε και νωρίτερα, είναι το αντικείμενο πρόγραμμα. Το πρόγραμμα αυτό είναι κατανοητό από τον υπολογιστή αφού είναι σε μορφή εντολών δυαδικής αναπαράστασης, όμως δεν μπορεί να εκτελεστεί. Για να γίνει η εκτέλεση του θα πρέπει να συνδεθεί με άλλα τμήματα προγράμματος, όπως με βιβλιοθήκες της συγκεκριμένης γλώσσας προγραμματισμού. Η έξοδος που παράγεται είναι το τελικό πρόγραμμα που πλέον μπορεί να εκτελεστή από τον υπολογιστή και ονομάζεται **εκτελέσιμο πρόγραμμα**.

Να σημειωθεί ότι η διαφορά μεταξύ μεταγλωττιστή και διερμηνευτή είναι κυρίως το μέγεθος του τμήματος κώδικα που μεταφράζουν. Ο μεταγλωττιστής μεταφράζει ολόκληρο το πηγαίο πρόγραμμα σε γλώσσα μηχανής, ενώ ο διερμηνευτής διαβάζει μία εντολή κάθε φορά του πηγαίου κώδικα και για κάθε μια εντολή παράγει την ισοδύναμη σε γλώσσα μηχανής.

Ο προγραμματισμός αποτελεί την θεμελιώδη διαδικασία σχεδίασης και ανάπτυξης λογισμικού. Βασίζεται σε μια σειρά από αρχές και έννοιες που είναι απαραίτητες για την επίλυση προβλημάτων μέσω του υπολογιστή. Στα βασικά στοιχεία του προγραμματισμού και συγκεκριμένα ενός προγράμματος, περιλαμβάνονται έννοιες όπως

- Μεταβλητές
- Τύποι δεδομένων
- Συνθήκες
- Βρόχοι
- Δομές δεδομένων
- Συναρτήσεις
- Αλγόριθμοι
- Εξφαλμάτωση / Αποσφαλμάτωση

Αυτά τα στοιχεία αποτελούν τον πυρήνα της προγραμματιστικής σκέψης, για την κατανόηση και την ανάπτυξη προγραμμάτων σε διάφορες γλώσσες προγραμματισμού. Παραδείγματα σύγχρονων γλωσσών προγραμματισμού είναι: C, C++, Java, Python, Html, JavaScript, Pascal, php και πολλές άλλες.

Αναφερθήκαμε νωρίτερα για τις γλώσσες προγραμματισμού υψηλού επιπέδου (high-level programming language). *«Ο όρος γλώσσα υψηλού επιπέδου υποδηλώνει ότι δεν είναι κατασκευασμένη για να λειτουργεί σε συγκεκριμένη αρχιτεκτονική υπολογιστή αλλά δύναται να λειτουργήσει σε πληθώρα αρχιτεκτονικών, γεγονός που σημαίνει ότι*

- *Είναι εύκολο να εκτελεσθεί σε οποιοδήποτε υπολογιστή με μικρές αλλαγές και όχι με επανασχεδιασμό του.*
- *Κάθε γλώσσα υψηλού επιπέδου χρειάζεται έναν μεταγλωττιστή ο οποίος μεταφράζει τις γραμμές ενός προγράμματος σε γλώσσα μηχανής ώστε να γίνεται κατανοητό το πρόγραμμα από τον υπολογιστή και να εκτελείται.»*

Οι γλώσσες προγραμματισμού υψηλού επιπέδου είναι σχεδιασμένες η κάθε μια τους για έναν διαφορετικό σκοπό. Η επιλογή της γλώσσας προγραμματισμού δεν είναι εύκολη και εξαρτάται από παράγοντες όπως το είδος του προγράμματος, τις γνώσεις και δεξιότητες του προγραμματιστή, την δυσκολία της γλώσσας. Αυτές διακρίνονται σε δύο μεγάλες κατηγορίες: **σε γλώσσες ειδικού σκοπού**, όπως η Fortran και σε **γλώσσες γενικής χρήσης**, όπως η Python ή η C.

2.1.α: ΙΣΤΟΡΙΚΗ ΑΝΑΔΡΟΜΗ

Η ιστορική αναδρομή του προγραμματισμού αποτελεί μια σημαντική πτυχή για την κατανόηση της εξέλιξης των γλωσσών προγραμματισμού.

Η ιστορία του προγραμματισμού ξεκινά τον 19ο αιώνα με πρώτη προγραμματίστρια μια γυναίκα, την Ada Lovelace. Έγραψε έναν αλγόριθμο, ένα «σχέδιο» σχετικά με τον τρόπο που ο υπολογιστής Αναλυτικής Μηχανής θα μπορεί να υπολογίζει αριθμούς Μπερνούλλι.

Αναφερθήκαμε νωρίτερα στις γλώσσες προγραμματισμού υψηλού επιπέδου. Αυτή η πληροφορία αυτόματα μας δημιουργεί μια απορία σχετικά με το αν υπάρχουν γλώσσες και χαμηλού επιπέδου. Η απάντηση σε αυτό είναι, φυσικά και υπάρχουν. Ποια λοιπόν η διαφορά;

Οι **γλώσσες χαμηλού επιπέδου** είναι φιλικές προς τον υπολογιστή. Τέτοιου είδους γλώσσες έχουν βασικές εντολές και λέξεις κλειδιά όπως όλες οι γλώσσες προγραμματισμού όμως πλησιάζουν περισσότερο στην νοοτροπία κατανόησης εντολών ενός υπολογιστή, καθώς είναι φτιαγμένες να είναι κατανοητές από αυτόν με χρήση του δυαδικού συστήματος. Έτσι, είναι πιο δύσκολο για έναν προγραμματιστή να τις κατανοήσει και επομένως και να τις χρησιμοποιήσει. Αυτή είναι και η κύρια αιτία δημιουργίας των γλωσσών υψηλού επιπέδου. Από την άλλη πλευρά, άρα, υπάρχουν οι γλώσσες υψηλού επιπέδου οι οποίες δεν είναι τόσο φιλικές προς τον υπολογιστή, αλλά περισσότερο ως προς τον προγραμματιστή. Παραδείγματα γλωσσών υψηλού επιπέδου, είναι η C++, η Java, η Pascal, η Python και η Visual Basic. Οι γλώσσες χαμηλού επιπέδου χρειάζονται περισσότερο χρόνο να εκτελεστούν, σε σχέση με τις γλώσσες υψηλού επιπέδου.

Η **Assembly** είναι μια από τις πρώτες γλώσσες προγραμματισμού χαμηλού επιπέδου, αναπτυγμένη στην δεκαετία του 1940-50. Βρίσκεται πιο κοντά στην γλώσσα μηχανής (binary code) και επιτρέπει στον προγραμματιστή να γράφει εντολές που μεταφράζονται απευθείας σε δυαδικό κώδικα. Αυτή η γλώσσα χρησιμοποιείται ακόμη και σήμερα από μερικούς προγραμματιστές, για την ανάπτυξη λογισμικού με υψηλή απόδοση όπως λειτουργικά συστήματα.

Μετά το 1950 εμφανίστηκαν οι πρώτες γλώσσες προγραμματισμού υψηλού επιπέδου για ευκολότερη ανάπτυξη αλγορίθμων. Η **Fortran** αναπτύχθηκε το ΄57 για επιστημονικούς υπολογισμούς. Ήταν η πρώτη γλώσσα που χρησιμοποιήθηκε ευρέως και αποτέλεσε την βάση για πολλές άλλες σύγχρονες γλώσσες.

Την δεκαετία του 1970 αναδύθηκε ο **αντικειμενοστραφής προγραμματισμός**. Αυτός βασίζεται στην έννοια των αντικειμένων όπως γνωρίζουμε από την καθημερινότητα, τα οποία αυτά αντικείμενα κατατάσσονται σε διάφορες κατηγορίες. Έτσι και τα αντικείμενα στον προγραμματισμό

ενσωματώνουν δεδομένα και μεθόδους. Ένα παράδειγμα αντικειμενοστραφούς γλώσσας είναι η **C++**, η οποία αναπτύχθηκε το ΄80 και αποτελεί επέκταση της πιο απλής σύγχρονης γλώσσας C, προσθέτοντας σε αυτήν την αντικειμενοστρέφεια.

Από το 1990 και έπειτα εξελίχθηκαν γλώσσες προγραμματισμού που χρησιμοποιούνται μέχρι και σήμερα. Οι **Java, Python, JavaScript** είναι μερικές από αυτές.

Η εξέλιξη του προγραμματισμού από τις πρώτες γλώσσες χαμηλού επιπέδου μέχρι τις σύγχρονες, δυναμικές γλώσσες αντικατοπτρίζει τη συνεχή προσπάθεια για πιο αποδοτική, κατανοητή και ισχυρή ανάπτυξη λογισμικού. Η ιστορική αναδρομή του προγραμματισμού παρέχει την βάση για την κατανόηση του πως οι σύγχρονες γλώσσες προγραμματισμού διαμορφώθηκαν και πως συνεχίζουν να εξελίσσονται σήμερα, για να καλύψουν τις ανάγκες του σύγχρονου κόσμου μας.

2.2: Η ΓΛΩΣΣΑ C

Στα πλαίσια της πτυχιακής εργασίας αναπτύχθηκε μια διαδικτυακή πλατφόρμα εκμάθησης της γλώσσας προγραμματισμού C. Πάμε να γνωρίσουμε την γλώσσα C και τα χαρακτηριστικά της.

Μια τεχνική σχεδίασης προγραμμάτων είναι ο αντικειμενοστραφής προγραμματισμός. Το 1960 παρουσιάστηκε μια άλλη τεχνική σχεδίασης, ο **δομημένος προγραμματισμός**. Ο όρος του δομημένου προγραμματισμού έχει να κάνει με την διαίρεση του προγράμματος σε επί μέρους μικρότερα τμήματα, τις συναρτήσεις. Κάθε συνάρτηση είναι ορισμένη να επιτελεί μια συγκεκριμένη ενέργεια και μπορεί να κληθεί είτε μέσα στο κύριο πρόγραμμα, είτε σε κάποιο άλλο από τα υπό προγράμματα του. Αναπτύχθηκε με σκοπό να δημιουργηθεί μια κοινή μεθοδολογία στην ανάπτυξη προγραμμάτων. Σήμερα είναι μια από τις βασικές μεθοδολογίες σύνταξης προγραμμάτων, που βοηθάει τον προγραμματιστή να αναπτύξει σύνθετα προγράμματα, βοηθάει στην ευκολότερη κατανόηση και επομένως επιδιόρθωση λαθών, κάτι το οποίο συμβάλει στην μείωση τους.

Η **γλώσσα C** είναι μια διαδικαστική γλώσσα με τις δυνατότητες του δομημένου προγραμματισμού. Τα προγράμματα της C πάντα αρχίζουν με την εκτέλεση των εντολών της κύριας συνάρτησης main και έπειτα αυτή μπορεί να καλέσει άλλες συναρτήσεις, οι οποίες και αυτές με την σειρά τους μπορούν να καλέσουν άλλες συναρτήσεις ή ακόμα και τον ίδιο τους τον εαυτό, επιτρέποντας έτσι την χρήση αναδρομής. Η C είναι μια γλώσσα προγραμματισμού γενικής χρήσης. Αναπτύχθηκε στα εργαστήρια της εταιρείας BELL και δημιουργήθηκε αρχικά για τα λειτουργικά συστήματα Unix. Αυτή, γνώρισε μεγάλη διάδοση. Είναι μια από τις πιο συνηθισμένες γλώσσες και αποτέλεσε την βάση για να δημιουργηθούν πολλές άλλες γλώσσες προγραμματισμού.

Η C αποτελεί μια δημοφιλή επιλογή γλώσσας για να ξεκινήσει κάποιος το ταξίδι εξερεύνησης και κατανόησης του προγραμματισμού και των δυνατοτήτων του. Είναι μια απλή σε επίπεδο κατανόησης γλώσσα στον κόσμο της πληροφορικής με ευρεία χρήση της σε εφαρμογές και λειτουργικά συστήματα. Αποτελεί μια λιτή γλώσσα για να μπορέσει κάποιος να εκπαιδευτεί στον προγραμματισμό, με πολλές λέξεις στα αγγλικά τις οποίες είναι εύκολο να κατανοήσεις την λειτουργία τους, με μικρό σε πλήθος αριθμό δεσμευμένων λέξεων (λέξεις-κλειδιά που η ονομασία τους δεν μπορεί να είναι για παράδειγμα το όνομα μιας μεταβλητής όπως θα δούμε και παρακάτω).

Αυτός είναι και ο βασικός λόγος που στην πτυχιακή εργασία αυτή η πλατφόρμα εκμάθησης επικεντρώνεται στην διδασκαλία της γλώσσας C. Η ιστοσελίδα που αναπτύχθηκε (θα αναφερθούμε και αργότερα για αυτό) απευθύνεται σε νέους προγραμματιστές και κυρίως σε μαθητές της δευτεροβάθμιας εκπαίδευσης. Αυτό σημαίνει ότι κατά την διάρκεια των «μαθημάτων» που υπάρχουν στην ιστοσελίδα δεν μπορεί να γίνει αναφορά σε όλες τις δυνατότητες της γλώσσας C. Η

πλατφόρμα αφορά εισαγωγικά μαθήματα τα οποία δεν επεκτείνονται πλήρως σε όλα τα κεφάλαια, αλλά παρέχει πληροφορίες σχετικά με τις βασικές αρχές και τις κύριες δυνατότητες της.

Γενικότερα όμως ας δούμε με λίγα λόγια τις δυνατότητες που έχει η γλώσσα C:

- Ελάχιστους περιορισμούς
- Ευελιξία
- Τμηματικό προγραμματισμό
- Λίγες αλλά ισχυρά δομημένες εντολές

Πρέπει πάντα να προσέχουμε κατά την σύνταξη των προγραμμάτων για τυχόν λάθη, γιατί οι λίγοι περιορισμοί της γλώσσας αυτής σημαίνει αυτόματα ότι υπάρχει και μικρότερος έλεγχος λαθών. Κατά τον χρόνο μεταγλώττισης, ο μεταγλωττιστής εντοπίζει τα συντακτικά λάθη (αν για παράδειγμα μια εντολή δεν ακολουθεί τους συντακτικούς κανόνες της γλώσσας) και στην συνέχεια δημιουργεί το εκτελέσιμο αρχείο κώδικα σε γλώσσα μηχανής, το οποίο εκτελείται άμεσα από τον υπολογιστή.

Τα στοιχεία που συνθέτουν ένα πρόγραμμα σε C είναι τέσσερα:

- Μεταβλητές
- Σταθερές
- Εκφράσεις
- Εντολές

Μεταβλητές: Για να μπορέσουμε να αναπαραστήσουμε και να αποθηκεύσουμε στο πρόγραμμα μας μια τιμή χρησιμοποιούμε μεταβλητές και σταθερές, όπως γίνεται και στα μαθηματικά. Και οι μεταβλητές και οι σταθερές είναι συμβολικά ονόματα τα οποία δέχονται τιμές, για να εισάγουμε δεδομένα στο πρόγραμμα καθώς και να αποθηκεύσουμε αποτελέσματα για να τα χρησιμοποιήσουμε- αν τα χρειαστούμε- στο μέλλον.

Οι ονομασίες και των σταθερών και των μεταβλητών αποθηκεύονται σε μια διεύθυνση - θέση μνήμης στον υπολογιστή. Μία θέση μνήμης στον υπολογιστή, είναι ένας αριθμός που αποτελεί και την διεύθυνση μνήμης. Οι διευθύνσεις αυτές είναι μοναδικές και επιλέγονται από τον υπολογιστή, με σκοπό να μπορεί να ανατρέξει σε κάποια μεταβλητή ανά πάσα στιγμή, όποτε το χρειαστεί.

Σε όλες τις μεταβλητές δεν αποθηκεύονται ίδιου τύπου τιμές. Αν θέλω εγώ ως προγραμματιστής να κρατήσω σε μια μεταβλητή την ηλικία μου για παράδειγμα έστω ότι είμαι 20 χρονών, τότε η τιμή αυτή είναι ακεραίου τύπου. Όλα τα διαφορετικά είδη μεταβλητών ονομάζονται τύποι δεδομένων (data types). Οι μεταβλητές πρέπει να δηλωθούν πριν από τη χρήση τους, με τον τύπο δεδομένων που θα αποθηκεύουν, και μπορεί να τους ανατεθεί μια αρχική τιμή κατά τη δήλωση τους.

Η C υποστηρίζει διάφορους τύπους δεδομένων, εκ των οποίων οι βασικοί είναι τρείς:

- **Ακέραιος** -> Αναφέρεται σε δεδομένα που είναι ακέραιοι αριθμοί και δηλώνεται με την λέξη int
- **Πραγματικός** -> Αναφέρεται σε δεδομένα που είναι πραγματικοί αριθμοί και δηλώνεται με τις λέξεις float ή double
- **Χαρακτήρας** -> Αναφέρεται σε μεμονωμένους χαρακτήρες και δηλώνεται με την λέξη char

```
int age;
age = 20;
ή
int age = 20;
```

```
float price;
price = 3.5;
double x = 7.2;
```

```
char name = 'A';
```

Σταθερές: Η διαφορά των σταθερών από τις μεταβλητές είναι ότι οι σταθερές είναι τιμές που δεν μπορούν να αλλάξουν κατά τη διάρκεια της εκτέλεσης του προγράμματος. Όπως συμβαίνει και στις μεταβλητές, έτσι και στις σταθερές οι τιμές που αποθηκεύονται μπορούν να είναι

διαφορετικών τύπων δεδομένων. Οι σταθερές μπορούν να είναι ακέραιοι, πραγματικοί, χαρακτήρες, ή οποιοσδήποτε άλλος τύπος δεδομένων υποστηρίζεται από τη C. Δηλώνονται με τη χρήση της δεσμευμένης λέξης **const**.

Εκφράσεις: Μια έκφραση - expression είναι πράξεις που έχουν ένα συγκεκριμένο αποτέλεσμα. Είναι ένας συνδυασμός μεταβλητών, σταθερών και τελεστών.

```
sum = x + y; // Άθροισμα
```

Εντολές: Κάθε εντολή είναι μια φράση της γλώσσας που υποχρεώνει τον υπολογιστή να εκτελέσει μια συγκεκριμένη λειτουργία. Αυτή η λειτουργία μπορεί να είναι μια απλή πράξη, ένας έλεγχος, μια επαναληπτική διαδικασία, μια εμφάνιση αποτελέσματος και πολλά άλλα. Ένα πρόγραμμα αποτελείται από μια ακολουθία εντολών, οι οποίες εκτελούνται με την σειρά από την αρχή του προγράμματος μέχρι το τέλος. Κάποιες εντολές στην C υπάρχουν ενσωματωμένες στην δομή της και άλλες υπάρχουν σε βιβλιοθήκες που υποστηρίζει η γλώσσα, με την πιο γνωστή να είναι η stdio.h με τις βασικότερες εντολές όπως η εισαγωγή ενός δεδομένου από το πληκτρολόγιο στο πρόγραμμα ή η εμφάνιση μιας πληροφορίας στην οθόνη.

```
printf("Hello everyone!");
```

Η C είναι μια γλώσσα προγραμματισμού η οποία πέρα από τα τέσσερα αυτά βασικά στοιχεία της υποστηρίζει και πολλές άλλες δυνατότητες που είναι χρήσιμες για έναν προγραμματιστή και κάνουν την διαδικασία σύνταξης ενός προγράμματος πολύ πιο εύκολη. Μερικές από αυτές είναι:

1. Διαχείριση μνήμης
   - **Δείκτες (pointers)** -> Παρέχει υποστήριξη για την διαχείριση μνήμης μέσω δεικτών επιτρέποντας έτσι την άμεση διαχείριση διευθύνσεων μνήμης. Αυτό είναι ιδιαίτερα βοηθητικό στις δομές δεδομένων και στις συναρτήσεις.
   - **Δυναμική κατανομή μνήμης** -> Μέσω έτοιμων συναρτήσεων (όπως malloc, calloc, realloc κ.α.) επιτρέπεται η δυναμική παραχώρηση μνήμης και η αποδέσμευση της κατά την εκτέλεση ενός προγράμματος. Αυτό είναι ιδιαίτερα βοηθητικό στην διαχείριση μεγάλου όγκου δεδομένων που δεν τον γνωρίζουμε εξαρχής και επιτρέπει την άμεση αλλαγή του μεγέθους για παράδειγμα σε μια δομή δεδομένων, κάτι που δεν μπορεί να επιτευχθεί με στατικά δεδομένα (όπως οι απλοί στατικοί πίνακες).
2. Υποστήριξη δομημένου προγραμματισμού
   - Το κύριο μέσο είναι ο τμηματικός προγραμματισμός που επιτυγχάνεται μέσω **συναρτήσεων**.
3. Επεξεργασία κειμένου και αλφαριθμητικών
   - Η C επιτρέπει την επεξεργασία κειμένου και αλφαριθμητικών μέσω συμβολοσειρών (string) και έχει βιβλιοθήκες με έτοιμες συναρτήσεις για την επεξεργασία τους. Σε άλλες γλώσσες προγραμματισμού τα **strings** θεωρούνται τύπος δεδομένων όμως η C δεν τον υποστηρίζει άμεσα. Δηλώνεται ως τύπος δεδομένων char σε συνδυασμό με μια δομή δεδομένων, τους πίνακες.
4. Δομές δεδομένων
   - Ένα από τα βασικότερα πλεονεκτήματα του προγραμματισμού είναι οι δομές δεδομένων για την μαζική τους διαχείριση. Η C υποστηρίζει τη δημιουργία σύνθετων δομών δεδομένων όπως **πίνακες**, **δομές** (structures), **λίστες**, **στοίβες** και **ουρές**. Οι πίνακες μπορούν να είναι είτε στατικοί, είτε δυναμικοί αναλόγως την δήλωση τους.

5. Αρχεία
   - *Κάθε οργανωμένη συλλογή δεδομένων που αποθηκεύεται σε ένα περιφερειακό μέσο αποθήκευσης, ονομάζεται αρχείο (file).* Επιτρέπει στον προγραμματιστή να διαβάζει και να γράφει δεδομένα από και προς αρχεία στο σύστημα. Τα αρχεία χρησιμοποιούνται για να αποθηκεύουν δεδομένα μόνιμα, ώστε να μπορούν να ανακτηθούν και να επεξεργαστούν αργότερα. Υπάρχουν τα αρχεία κειμένου (text files) και τα δυαδικά αρχεία (binary files).

Δομή ενός προγράμματος σε C:

Τα πάντα ξεκινούν από την συνάρτηση main() που αναφέραμε και νωρίτερα, η οποία συνοδεύεται με αγκύλες όπου μέσα σε αυτές υπάρχουν όλες οι εντολές του προγράμματος. Αυτή είναι η πρώτη που εκτελείται στο πρόγραμμα και πρέπει πάντα σε κάθε πρόγραμμα να υπάρχει μόνο μία τέτοια συνάρτηση. Στην C κάθε συνάρτηση έχει την δυνατότητα να επιστρέφει μόνο ένα δεδομένο. Το δεδομένο που επιστρέφει μπορεί να είναι για παράδειγμα μια μεταβλητή. Ότι τύπο δεδομένων έχει το δεδομένο που επιστρέφεται με την εντολή return, τον ίδιο τύπο δεδομένων θα είναι και η συγκεκριμένη συνάρτηση. Για παράδειγμα:

```
int main() {
    // υπόλοιπες εντολές

    return 0;
}
```

Αν δεν θέλω μια συνάρτηση να επιστρέφει κάποια τιμή τότε αυτή θα είναι τύπου void. Όταν ο υπολογιστής "διαβάσει" την εντολή return τότε κατευθείαν θα τερματίσει το πρόγραμμα, αγνοώντας οποιαδήποτε εντολή υπάρχει μετά από αυτήν. Τι θα επιστρέψει το πρόγραμμα, το ορίζει ο προγραμματιστής του -δηλαδή εμείς. Όταν όλα βαίνουν καλώς και το πρόγραμμα μας εκτελείται κανονικά τότε μπορούμε να προσθέσουμε στο τέλος του προγράμματος μας την εντολή

```
return 0;
```

και επειδή επιστρέφεται η τιμή 0 τότε η συνάρτηση main θα είναι ακεραίου τύπου δεδομένων, δηλαδή int.

- Κάθε εντολή θα πρέπει να συνοδεύεται στο τέλος της με το σύμβολο του ελληνικού ερωτηματικού (;)
- Τα σχόλια μονής γραμμής στην γλώσσα αναπαρίστανται με //

Ένα πολύ απλό πρώτο παράδειγμα σε C το οποίο εκτυπώνει στην οθόνη μας την φράση «Hello everyone!» είναι το εξής:

```
#include <stdio.h>

int main(){
    printf("Hello everyone!");
    return 0;
}
```

Πάντα με την συνοδεία της βασικής βιβλιοθήκης. Όταν η συνάρτηση main() δεν έχει κάποια παράμετρο μέσα στις παρενθέσεις τότε απλά λέμε ότι δεν δέχεται ορίσματα. Ως παράμετρος θα μπορούσε να είναι για παράδειγμα μια μεταβλητή.

Ένα πρόγραμμα με περισσότερες από μια συναρτήσεις δείχνει κάπως έτσι:

```
#include <stdio.h>
void show();    //δεν έχει παραμέτρους

int main(int argc, char *argv []){
    show();

    return 0;
}

void show(){
    printf ("hello world!\n");
}
```

Η γλώσσα C, με τις ισχυρές δυνατότητές της, υπήρξε θεμέλιο για την ανάπτυξη πολλών σύγχρονων γλωσσών προγραμματισμού και την καθιστούν κατάλληλη για την εκπαίδευση νέων προγραμματιστών. Η απλότητά της, συνδυασμένη με ισχυρά χαρακτηριστικά, μερικά από αυτά κοινά με την Pascal, αλλά και με πολλές δυνατότητες γλώσσας χαμηλού επιπέδου και τον δομημένο προγραμματισμό, την καθιστούν κατάλληλη για μια ευρεία γκάμα εφαρμογών. Η εξέλιξή της ήρθε την δεκαετία του '70 με τον εμπλουτισμό της γλώσσας με αντικειμενοστραφή χαρακτηριστικά. Η γλώσσα αυτή εξελίχθηκε στην C++ διευρύνοντας τις δυνατότητές της για ανάπτυξη πιο σύνθετων και ευέλικτων εφαρμογών.

2.3: ΕΦΑΡΜΟΓΕΣ ΕΚΜΑΘΗΣΗΣ ΓΛΩΣΣΩΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

Τίτλος της πτυχιακής εργασίας αυτής είναι: **ΑΝΑΠΤΥΞΗ ΔΙΑΔΙΚΤΥΑΚΗΣ ΠΛΑΤΦΟΡΜΑΣ ΓΙΑ ΕΙΣΑΓΩΓΗ ΣΤΟΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ ΣΤΗΝ ΔΕΥΤΕΡΟΒΑΘΜΙΑ ΕΚΠΑΙΔΕΥΣΗ**. Το θέμα λοιπόν αυτό πραγματεύεται γύρο από την εκμάθηση γλωσσών προγραμματισμού και συγκεκριμένα -όπως αναφέραμε και νωρίτερα- για την γλώσσα C. Υπάρχουν διάφορα μέσα που μπορεί να χρησιμοποιήσει ένας νέος προγραμματιστής για να διευρύνει τις γνώσεις του σε αυτόν τον τομέα.

Οι εφαρμογές εκμάθησης γλωσσών προγραμματισμού είναι εργαλεία, τα οποία βοηθούν τους χρήστες να εκπαιδευτούν στην ανάπτυξη προγραμμάτων, μέσω διαδραστικών μεθόδων, πρακτικών ασκήσεων και ποικίλο εκπαιδευτικό υλικό. Τέτοιες εφαρμογές είναι σχεδιασμένες για να υποστηρίζουν προγραμματιστές σε διάφορα επίπεδα, από αρχάριους μέχρι πιο προχωρημένους προγραμματιστές και να προσφέρουν ένα περιβάλλον που όχι μόνο μπορούν να πάρουν γνώσεις, αλλά και να τις εξασκήσουν.

Στην εποχή που διανύουμε, η τεχνολογική εξέλιξη έχει συμβάλει και έχει επηρεάσει δραστικά την καθημερινότητα όλων μας. Η εκμάθηση γλωσσών προγραμματισμού έχει γίνει μια αρκετά θα λέγαμε «αναγκαία» δεξιότητα για μαθητές που ενδιαφέρονται να εξερευνήσουν τον κόσμο της πληροφορικής. Η πληθώρα διαθέσιμων πόρων επιτρέπει σε νέους και νέες να ξεκινήσουν μια πορεία στον προγραμματισμό από νωρίς, ακόμα και κατά την διάρκεια φοίτησης των σπουδών τους σε δευτεροβάθμια κλίμακα ή και μετέπειτα. Η εύκολη στις μέρες μας πρόσβαση στην τεχνολογία και στο ίντερνετ έχει διευρυνθεί και η διαδικτυακή μάθηση αποτελεί πλέον μια κύρια πηγή γνώσεων.

Σε αυτό το κεφάλαιο θα μιλήσουμε για τις διαδικτυακές μεθόδους εκμάθησης γλωσσών προγραμματισμού και των βασικών εργαλείων που είναι διαθέσιμα.

i. Πλατφόρμες για κινητές συσκευές:

Η εκμάθηση μέσω κινητών συσκευών έχει γίνει εξαιρετικά δημοφιλής λόγω της εύκολης χρήσης και της άμεσης πρόσβασης που προσφέρει, ιδιαίτερα σε νέους. Υπάρχουν πλεονεκτήματα και μειονεκτήματα στην εκμάθηση γλωσσών μέσω κινητών συσκευών σε σύγκριση με έναν υπολογιστή.

Τα πλεονεκτήματα που προσφέρουν οι κινητές συσκευές έχουν να κάνουν με:

- Την εύκολη πρόσβαση στην συσκευή -> Πολλές εφαρμογές για κινητά προσφέρουν μικρά, διαδραστικά μαθήματα που είναι εύκολα κατανοητά και μπορείς να τα ολοκληρώσεις σε λίγα λεπτά.
- Την εύκολη φορητότητα -> Οι εφαρμογές για κινητά επιτρέπουν την εκμάθηση από οποιοδήποτε σημείο, οποτεδήποτε, με ευκολία. Μπορείς να κάνεις εξάσκηση σε μικρά διαστήματα, όπως όταν περιμένεις σε μια ουρά ή χρησιμοποιείς τα μέσα μεταφοράς.

Από την άλλη μεριά τα μειονεκτήματα έχουν να κάνουν με:

- Το περιορισμένο περιβάλλον προγραμματισμού -> Οι κινητές συσκευές έχουν μικρότερες οθόνες και περιορισμένους πόρους (όπως επεξεργαστική ισχύ).
- Λιγότερες δυνατότητες -> Η χρήση κινητών συσκευών για εκμάθηση γλωσσών προγραμματισμού συνήθως παρέχει λιγότερη δυνατότητα επεξεργασίας για παράδειγμα του κώδικα μας και λιγότερη ευχρηστία για σύνθετα έργα, σε σύγκριση με έναν υπολογιστή.

Γενικότερα, όπως αναφέρει και μια έρευνα σε ένα σχετικό άρθρο του Near East University της Κύπρου: «*Η ανάπτυξη εφαρμογών για κινητές συσκευές απαιτεί την εξέταση των περιορισμών και των χαρακτηριστικών αυτών των συσκευών. Οι κινητές συσκευές λειτουργούν με μπαταρία και έχουν λιγότερο ισχυρούς επεξεργαστές από τους προσωπικούς υπολογιστές, αλλά διαθέτουν περισσότερες δυνατότητες, όπως η ανίχνευση τοποθεσίας και οι κάμερες. Οι προγραμματιστές πρέπει επίσης να λάβουν υπόψη τους μια ευρεία γκάμα μεγεθών οθόνης, προδιαγραφών υλικού και διαμορφώσεων λόγω του έντονου ανταγωνισμού στο λογισμικό κινητών συσκευών και των αλλαγών σε κάθε πλατφόρμα… Η ανάπτυξη εφαρμογών για κινητά απαιτεί τη χρήση εξειδικευμένων ολοκληρωμένων περιβαλλόντων ανάπτυξης. Οι εφαρμογές για κινητά δοκιμάζονται αρχικά στο περιβάλλον ανάπτυξης χρησιμοποιώντας εξομοιωτές και στη συνέχεια υποβάλλονται σε δοκιμές πεδίου.*»

Μια γνωστή εφαρμογή εκμάθησης γλωσσών για κινητές συσκευές είναι το **SoloLearn** που επιτρέπει σε μαθητές να μάθουν προγραμματισμό από το κινητό τους τηλέφωνο ή tablet, προσφέροντας διαδραστικά μαθήματα, κουίζ κ.α. Επιπλέον υπήρχε και το **TouchDevelop**. Το TouchDevelop αρχικά σχεδιάστηκε για κινητές συσκευές και η χρήση του επικεντρώθηκε σε αυτές. Ωστόσο, με την πάροδο του χρόνου έγινε προσβάσιμο και μέσω υπολογιστών μέσω ενός διαδικτυακού περιβάλλοντος. Ήταν ένα εργαλείο ανάπτυξης εφαρμογών που δημιουργήθηκε από τη Microsoft Research και είχε σχεδιαστεί για να επιτρέπει στους χρήστες να δημιουργούν εφαρμογές απευθείας από μια φορητή συσκευή, όπως ένα smartphone ή tablet. Αντίθετα με τα παραδοσιακά περιβάλλοντα ανάπτυξης, το TouchDevelop ήταν ειδικά σχεδιασμένο για χρήση σε οθόνες αφής, κάτι που το έκανε ιδιαίτερα εύχρηστο για κινητές συσκευές. Η Microsoft διέκοψε την υποστήριξη του το 2019, και πλέον δεν είναι διαθέσιμο για χρήση. Ωστόσο, το εργαλείο αυτό ήταν σημαντικό στην εκπαίδευση προγραμματισμού. Το TouchDevelop εξελίχθηκε στο **MakeCode**, ένα από τα εκπαιδευτικά εργαλεία της Microsoft που εστιάζει στη διδασκαλία του προγραμματισμού με έναν διασκεδαστικό και διαδραστικό τρόπο. Είναι κυρίως σχεδιασμένο για χρήση σε υπολογιστές, αλλά μπορεί να λειτουργήσει και σε κινητές συσκευές μέσω του προγράμματος περιήγησης.

ii. Εκπαιδευτικά Βίντεο:

Τα εκπαιδευτικά βίντεο έχουν αναδειχθεί σε ένα από τα πιο δημοφιλή εργαλεία μάθησης προγραμματισμού. Όλοι οι νέοι και οι έμπειροι προγραμματιστές κάποια στιγμή στην ζωή μας έχουμε βρεθεί στην θέση να ψάξουμε μια πληροφορία για ένα πρόβλημα στον κώδικα μας και η λύση να υπάρχει σε ένα αντίστοιχο βίντεο ενός δημιουργού Έλληνα και μη. Κανάλια στο YouTube ασχολούνται με την διδασκαλία γλωσσών μέσα από πλήρης σειρές από tutorials, τα οποία καλύπτουν ένα ευρύ φάσμα πληροφοριών. Ένας Έλληνας έμπειρος προγραμματιστής που μεταδίδει τις γνώσεις του μέσα από βίντεο είναι ένα κανάλι στο YouTube με όνομα **CodeGRow**. Επιπλέον, πλατφόρμες όπως το **Khan Academy** προσφέρουν βίντεο-μαθήματα που καλύπτουν τόσο τα βασικά όσο και τα προχωρημένα θέματα στον προγραμματισμό.

iii. Διαδικτυακές Πλατφόρμες Συνομιλίας:

Υπάρχουν στο ίντερνετ αρκετές κοινότητες που προσφέρουν υποστήριξη και βοήθεια σε προγραμματιστές που αντιμετωπίζουν κάποιο πρόβλημα είτε στον κώδικα, είτε σε κάποιο σφάλμα, είτε στο setup εφαρμογών και πολλά άλλα. Μέσω τέτοιων πλατφορμών έχουμε την δυνατότητα να κάνουμε ερωτήσεις και να λαμβάνουμε διαφορετικές απαντήσεις από άλλους χρήστες σε μορφή συνομιλίας, οι οποίοι έχουν αντιμετωπίσει παρόμοιο ή και το ίδιο πρόβλημα με εμάς στο παρελθόν. Αυτό κάνει την επικοινωνία και την αλληλεπίδραση με άτομα ίδιας «κουλτούρας» πιο άμεση.

Μία πλατφόρμα που επιδιώκει ακριβώς αυτό είναι το **Stack Overflow**, που όσα άτομα ασχολούνται με τον προγραμματισμό σίγουρα θα συναντήσουν μπροστά τους.

iv. Διαδικτυακές Πλατφόρμες Εκμάθησης Προγραμματισμού:

Οι διαδικτυακές πλατφόρμες εκμάθησης είναι το βασικό μέσω διαδικτυακής εκπαίδευσης γλωσσών προγραμματισμού. Προσφέρουν πρόσβαση σε μαθήματα κυρίως μέσω του υπολογιστή αλλά και από κινητό, καθιστώντας την εκπαίδευση προσιτή και ευέλικτη. Η μεγαλύτερη σε πλήθος γκάμα εφαρμογών εκμάθησης γλωσσών είναι διαδικτυακές πλατφόρμες, οι οποίες προσφέρουν μια σειρά μαθημάτων (online courses) που καλύπτουν από τις βασικές αρχές του προγραμματισμού, μέχρι προχωρημένες τεχνικές. Οι μαθητές έχουν την δυνατότητα να επιλέξουν από μια μεγάλη ποικιλία γλωσσών προγραμματισμού όπως Python, Java, C με το μεγαλύτερο πλεονέκτημα να είναι αυτό της κατανομής του χρόνου τους ανάλογα με τα κριτήρια του καθενός. Ο κάθε ένας μας μπορεί να προσαρμόσει τα μαθήματα αυτά στον δικό του ρυθμό.

Σήμερα, η διαδικτυακή εκπαίδευση προσφέρει μια ευέλικτη εναλλακτική λύση για την παρακολούθηση μαθημάτων, επιτρέποντας την εκμάθηση από το σπίτι χωρίς τους περιορισμούς της παραδοσιακής πανεπιστημιακής εκπαίδευσης. Αντί για πρωινές ώρες και φυσική παρουσία, έχουμε πρόσβαση σε μαθήματα με ηρεμία και άνεση. Τα διαδικτυακά μαθήματα συνήθως περιλαμβάνουν πληροφορίες για το μάθημα, χρονοδιάγραμμα, πίνακα ανακοινώσεων, χάρτη προγράμματος σπουδών, διδακτικό υλικό (όπως άρθρα και διαφάνειες), δυνατότητες επικοινωνίας μέσω φόρουμ και email, καθώς και εργαλεία διαχείρισης φοιτητών για αξιολογήσεις και παρακολούθηση προόδου, ενώ παρέχουν επίσης συνδέσμους σε χρήσιμους εσωτερικούς και εξωτερικούς ιστοτόπους.

Ένας τύπος διαδικτυακού μαθήματος που προσφέρεται μέσω διαφορετικών πλατφορμών είναι τα **MOOC**. Είναι μαθήματα που είναι ανοιχτά και διαθέσιμα σε μεγάλο αριθμό ατόμων μέσω του διαδικτύου και αποτελούν μια καλή επιλογή για μαζική εκπαίδευση σε γλώσσες προγραμματισμού, σχεδιασμένα για διαφορετικά επίπεδα εκπαίδευσης. Προσφέρουν διαφορετικά επίπεδα πιστοποίησης, ανάλογα με την πλατφόρμα παρακολούθησης τους και το μάθημα επιλογής. Επιπλέον, μία Ουκρανική έρευνα ενός πανεπιστημίου με τίτλο: *«Using MOOC to Learn the Python Programming Language*» του 2023 αναφέρει ότι στόχος των MOOC είναι να απευθυνθεί σε:

- *«Students who study additional material within a specific discipline.*
- *Working people who want to improve their qualifications in their spare time.*
- *Users who wish to change their own direction of professional activity, etc.»*

Κατά βάση υπάρχουν δύο είδη διαδικτυακών πλατφορμών. Οι εφαρμογές που προσφέρουν κάποιο είδος πιστοποίησης μετά την ολοκλήρωση των μαθημάτων και οι εφαρμογές που απλά προσφέρουν γνώσεις.

- ✓ Παραδείγματα εφαρμογών με πιστοποίηση είναι αυτές που προσφέρονται σε μορφή ενός σεμιναρίου και συχνά συνοδεύονται από κάποια χρηματική συνεισφορά. Το **Udemy** είναι μια από τις πιο δημοφιλείς πλατφόρμες όπου οι μαθητές μπορούν να επιλέξουν και να αγοράσουν σεμινάρια μέσα από μια μεγάλη ποικιλία μαθημάτων, να παρακολουθήσουν βίντεο-μαθήματα και να συμμετάσχουν σε διαδραστικές ασκήσεις. Μετά το πέρας των μαθημάτων, οι συμμετέχοντες λαμβάνουν πιστοποιητικά ολοκλήρωσης που μπορούν να προσθέσουν στο βιογραφικό τους. Το **Coursera** είναι ένα ακόμη παράδειγμα μιας ίδιας σε φιλοσοφία ιστοσελίδας. Αυτό συνεργάζεται με κορυφαία πανεπιστήμια και εκπαιδευτικά ιδρύματα για την προσφορά μαθημάτων και πιστοποιητικών σε διάφορες τεχνολογίες και γλώσσες προγραμματισμού. Τα σεμινάρια συχνά περιλαμβάνουν εργασίες, εξετάσεις και project, επιτρέποντας στους μαθητές να εφαρμόσουν τις γνώσεις τους πρακτικά. Η ολοκλήρωση μαθημάτων συνοδεύεται από επίσημα πιστοποιητικά.

- ✓ Παραδείγματα εφαρμογών που δεν προσφέρουν κάποια πιστοποίηση είναι συνήθως και αυτά που δεν χρειάζεται ο προγραμματιστής να αγοράσει. Αυτές οι πλατφόρμες παρέχουν δωρεάν ή χαμηλού κόστους εκπαιδευτικό υλικό και επικεντρώνονται στη διαδραστική εκμάθηση. Είναι ιδανικές για αρχάριους που επιθυμούν να αποκτήσουν βασικές γνώσεις και εμπειρίες χωρίς να χρειάζεται να επενδύσουν σε πληρωμένα μαθήματα. Το **W3Schools**, το **GeeksforGeeks** και το **Javatpoint** προσφέρουν ένα ευρύ εκπαιδευτικό υλικό σε γλώσσες όπως Html, JavaScript, SQL και πολλά ακόμη. Εστιάζουν στην παράθεση του θεωρητικού πλαισίου της γλώσσας, σε tutorial, παραδείγματα κώδικα και ασκήσεις.

Η εκμάθηση γλωσσών προγραμματισμού έχει γίνει πιο προσιτή από ποτέ, χάρη στη διαθεσιμότητα των διαδικτυακών πόρων και εργαλείων. Οι μαθητές μπορούν να ξεκινήσουν την εκπαίδευσή τους μέσω διαδικτυακών μαθημάτων, εφαρμογών, εκπαιδευτικών βίντεο και διαδραστικών περιβαλλόντων, ενώ έχουν τη δυνατότητα να συμμετέχουν σε διαγωνισμούς που θα ενισχύσουν τις δεξιότητές τους. Με αυτόν τον τρόπο, η διαδικτυακή εκμάθηση προγραμματισμού ανοίγει νέους ορίζοντες για νέους και νέες που θέλουν να εξερευνήσουν τον κόσμο της τεχνολογίας και της πληροφορικής.

ΚΕΦΑΛΑΙΟ 3 ΤΕΧΝΟΛΟΓΙΕΣ ΑΝΑΠΤΥΞΗΣ ΕΦΑΡΜΟΓΩΝ

3.1: ΔΙΑΘΕΣΙΜΕΣ ΤΕΧΝΟΛΟΓΙΕΣ

Αναλύσαμε πολλούς διαφορετικούς τρόπους με τους οποίους μπορεί να γίνει η εκμάθηση γλωσσών προγραμματισμού. Στην εργασία αυτή επιλέχθηκε ο τελευταίος από τους τρόπους που αναλύθηκαν στο προηγούμενο κεφάλαιο. Πιο συγκεκριμένα, η εκμάθηση μέσω διαδικτυακών εφαρμογών. Ένα από τα ερωτήματα που προκύπτουν είναι**: Πως όμως μπορεί να υλοποιηθεί ένα τέτοιο project;**

Η ανάπτυξη διαδικτυακών πλατφορμών εκμάθησης γλωσσών προγραμματισμού απαιτεί την χρήση σύγχρονων τεχνολογιών, οι οποίες προσφέρουν ευελιξία, διαδραστικότητα και αλληλεπίδραση με τον χρήστη της εφαρμογής. Υπάρχουν αρκετές τεχνολογίες που επικεντρώνονται στην ίδια λειτουργία, απλά προσφέρονται με διαφορετικό τρόπο και ο προγραμματιστής είναι αυτός που θα επιλέξει την τεχνολογία που είναι πιο κατάλληλη και εύκολη στην χρήση για αυτόν και την εφαρμογή του. Οι διαθέσιμες τεχνολογίες καλύπτουν ένα ευρύ φάσμα, από τα εργαλεία που χρησιμοποιούνται για την ανάπτυξη ιστοσελίδων -με τις βασικότερες να είναι Html, CSS, JavaScript- έως και πιο εξελιγμένες λύσεις για την εκτέλεση κώδικα -όπως Node.js και APIs για αποστολή και επεξεργασία δεδομένων. Κάθε ολοκληρωμένη σύγχρονη πλατφόρμα συνδυάζει μερικές από τις τεχνολογίες αυτές μαζί και δεν χρησιμοποιεί μόνο μία κάθε φορά, ώστε να παρέχει ένα φιλικό περιβάλλον για τους χρήστες.

Κατά βάση οι τεχνολογίες ανάπτυξης εφαρμογών μπορούν να κατηγοριοποιηθούν σε δύο βασικά πεδία, τις τεχνολογίες **frontend** και τις τεχνολογίες **backend**. Οι τεχνολογίες αυτές εξυπηρετούν διαφορετικές ανάγκες, με το frontend να επικεντρώνεται στη διεπαφή χρήστη και το backend στη διαχείριση των δεδομένων και της λογικής της εφαρμογής. Οι τεχνολογίες αυτές, σημαντικό να τονιστεί ότι, είναι χρήσιμο να συνεργάζονται μεταξύ τους για να βγει ένα καλό αποτέλεσμα.

Στο παρόν κεφάλαιο, θα αναλυθούν οι διαθέσιμες τεχνολογίες που μπορεί να χρησιμοποιήσει ένας προγραμματιστής για την ανάπτυξη μιας εκπαιδευτικής πλατφόρμας προγραμματισμού. Θα παρουσιαστούν διάφορες εναλλακτικές προσεγγίσεις, τόσο για την ανάπτυξη του frontend όσο και του backend, καθώς και τα εργαλεία και πλατφόρμες που μπορούν να αξιοποιηθούν για τη φιλοξενία και την εκτέλεση του κώδικα.

3.1.α: ΤΕΧΝΟΛΟΓΙΕΣ FRONTEND

Οι τεχνολογίες του frontend όπως ίσως μπορούμε να καταλάβουμε και από το όνομα τους, ασχολούνται με τον κώδικα της εφαρμογής ώστε να σχηματιστεί το περιβάλλον της ιστοσελίδας και το τελικό αποτέλεσμα που είναι ορατό στον χρήστη. Για παράδειγμα το περιεχόμενο της ιστοσελίδας, τι αναγράφεται δηλαδή μέσα σε αυτήν ή η μορφοποίηση της γραμματοσειράς του κειμένου, ένα μενού και πολλά άλλα είναι αυτά τα οποία φτιάχνονται με τεχνολογίες frontend και είναι αυτά τα στοιχεία στα οποία έχει πρόσβαση ο χρήστης. Στην προκειμένη περίπτωση ο χρήστης της ιστοσελίδας μας είναι ένας μαθητής ή ένας νέος προγραμματιστής.

Το «τρίπτυχο της επιτυχίας» για την ανάπτυξη ιστοσελίδων βασίζεται στα θεμέλια από τρεις γλώσσες προγραμματισμού οι οποίες συνήθως συνδυάζονται μεταξύ τους. Ο λόγος γίνεται για τις **Html**, **CSS**, **JavaScript**.

- ✓ **HTML:**

Η γλώσσα HTML (Hypertext Markup Language) είναι υπεύθυνη για την δημιουργία της δομής της ιστοσελίδας. Είναι το βασικότερο μέλος στην παρέα μας και χρησιμοποιείται πάντα σε κάθε διαδικτυακή εφαρμογή.

Η ιστορία της ξεκινά την δεκαετία του 1990 από τον Tim Berners-Lee, έναν φυσικό του ερευνητικού κέντρου CERN. Το 1991 παρουσίασε το πρωτότυπο της Html με κείμενα και παραγράφους. Το ’93 έγινε η αναβάθμιση της στην Html 2 με εισαγωγή νέων επιπλέον χαρακτηριστικών και βελτιώσεων στην μορφοποίηση των κειμένων. Η Html 3 έφερε μεγαλύτερη ευελιξία στοιχείων και ενσωμάτωσε πολυμέσα, όπως εικόνες. Μετά δημιουργήθηκε η έκδοση 4. Αυτή με την σειρά της, είχε βελτιώσεις στην μορφοποίηση της ιστοσελίδας και υποστήριζε για πρώτη φορά την γλώσσα CSS. Σήμερα υπάρχει το **Html 5** που δημοσιεύθηκε τον Οκτώβριο του 2014, με υποστήριξη νέων πάλι τεχνολογιών -όπως την υποστήριξη βίντεο ή ήχων- με φόρμες, αποθήκευση δεδομένων κ.α.

Τα αρχεία html έχουν την κατάληξη **.html** συνοδευόμενη από το όνομα του αρχείου. Η γλώσσα αυτή περιέχει μια σειρά στοιχείων που ονομάζονται tags, τα οποία καθορίζουν το περιεχόμενο που πρόκειται να εμφανιστεί στην οθόνη, δηλαδή πως τα δεδομένα θα εμφανιστούν στο πρόγραμμα περιήγησης. Αυτά τα tags μπορούν να περιέχουν κείμενα, εικόνες, φόρμες, πίνακες, συνδέσμους, κουμπιά κ.α. Οι περισσότερες από αυτές τις ετικέτες έρχονται σε ζεύγη. Συγκεκριμένα, ένα tag συνήθως «ανοίγει» και «κλείνει».

Πάμε να δούμε μερικά από τα πιο συνηθισμένα:

<!DOCTYPE html> ⟶ Τοποθετείτε στην αρχή κάθε αρχείου για να δηλώσει ότι η έκδοση της γλώσσας είναι η πιο πρόσφατη (HTML5).

<html> … </html> ⟶ Η ετικέτα αυτή περιλαμβάνει όλο το περιεχόμενο της ιστοσελίδας. Μέσα σε αυτό τοποθετούνται όλα τα υπόλοιπα tags.

<head> … </head> ⟶ Εδώ τοποθετούνται δεδομένα όπως ο τίτλος της σελίδας, άλλα αρχεία που θέλουμε να συνδέσουμε με το html αρχείο κ.α. Αυτό το τμήμα δεν είναι ορατό στον χρήστη άμεσα.

<body> … </body> ⟶ Το body είναι το ορατό κομμάτι της ιστοσελίδας που βλέπει ο χρήστης. Μέσα σε αυτό ορίζονται οι ετικέτες με το κείμενο, τις παραγράφους ,τις εικόνες κ.α.

<h1> … </h1> ⟶ Η ετικέτα αυτή δηλώνει την επικεφαλίδα. Τα tags των επικεφαλίδων έχουν εύρος από h1 μέχρι h6, με το 1 να είναι η μεγαλύτερη σε μέγεθος γραμματοσειρά και 6 η μικρότερη.

*** … *** ⟶ Αυτό ορίζει τις παραγράφους για ένα κείμενο.

<img> ⟶ Χρησιμοποιείται για την εισαγωγή εικόνων στην σελίδα. Αυτό το tag είναι από τα λίγα που δεν χρειάζονται να κλείσουμε όταν γράφουμε το αρχείο. Αρκεί να καθορίσουμε μια παράμετρο, το src. Το source είναι η πηγή, δηλαδή η εικόνα (.jpg , .png) και ορίζεται ως εξής: **<img src=”path_to_image/image.jpg”>**

***<a> … *** ⟶ Το a είναι η ετικέτα για τους υπέρ-συνδέσμους ή αλλιώς links που συνδέουν το σάιτ μας με άλλες ιστοσελίδες. Η σύνταξη της περιλαμβάνει μια παράμετρο το href το οποίο ορίζεται ως εξής: ** Αυτός είναι ένας σύνδεσμος **

***
*** ⟶ Άλλη μια ετικέτα χωρίς κλείσιμο είναι αυτή που δηλώνει την αλλαγή γραμμής σε μια παράγραφο του κειμένου.

*** … *** ⟶ Το div χρησιμοποιείται για να ομαδοποιεί τμήματα (για παράδειγμα μια παράγραφο) και εμφανίζει συνήθως την μορφοποίηση που έχει υποστεί το κείμενο μέσω ενός αρχείου CSS.

*** … *** ⟶ Αντίθετα με το div, το span εκτελεί την ίδια λειτουργία σε μικρότερα τμήματα κειμένου μέσα σε παραγράφους (για παράδειγμα μια πρόταση).

Γενικότερα, η html υποστηρίζει πολλές λειτουργίες όπως εισαγωγή δεδομένων μέσω form, έντονη μορφοποίηση κειμένου (bold), εισαγωγή κουμπιών (buttons) και πολλά ακόμη.

Εδώ είναι ένα πρώτο παράδειγμα σε γλώσσα Html που συνδυάζει μερικές από τις δυνατότητες της.

```
<!DOCTYPE html>
<html>

<head>
  <title>Παράδειγμα σε HTML</title>
</head>

<body>
  <h1>Αυτή είναι η πρώτη μου ιστοσελίδα.</h1>

  <div>
    <h5>Λίγα λόγια:</h5>
    <p>Αυτή είναι μια <span>παράγραφος</span> με <b>έντονο</b> κείμενο.</p>

    <br>
    <img src="image.jpg">
    <br>

    <p>Για περισσότερες πληροφορίες, επισκεφθείτε το παρακάτω link:</p>
    <a href="https://example.com">Πάτησε εδώ </a>

  </div>

</body>
</html>
```

- ✓ **CSS:**

Τα αρχικά CSS προέρχονται από την ονομασία Cascading Style Sheets. Η γλώσσα αυτή είναι μια γλώσσα μορφοποίησης κειμένου, εικόνων, παραγράφων, κουμπιών, φορμών και γενικότερα οποιασδήποτε λειτουργίας υποστηρίζει η Html. Έκανε την εμφάνιση της στην έκδοση 4 της Html και είναι ιδιαίτερα βοηθητική διότι ο προγραμματιστής μπορεί να ομαδοποιήσει κάποια στοιχεία τα οποία εμφανίζονται στην οθόνη του χρήστη, ορίζοντας την μορφοποίηση που θέλει μια μόνο φορά και έπειτα όλα τα ίδια αυτά στοιχεία πρόκειται να εμφανίζονται με τον ίδιο τρόπο στην οθόνη.

Το αρχείο της γλώσσας αυτής μπορεί να συνδεθεί με το αρχείο της Html με δύο τρόπους. Ο πρώτος τρόπος είναι στο αρχείο Html να προσθέσουμε την ετικέτα **<style> … </style>** μέσα στην ήδη υπάρχουσα ετικέτα head. Επιπλέον, το style μπορεί να προστεθεί και σε ένα συγκεκριμένο αντικείμενο του κειμένου όπως για παράδειγμα μια επικεφαλίδα. Αν δεν θέλουμε να αλλάξουμε την μορφοποίηση όλων των επικεφαλίδων που υπάρχουν στην ιστοσελίδα, αλλά θέλουμε να αλλάξουμε το χρώμα πχ σε μια μόνο συγκεκριμένη κεφαλίδα τότε χρησιμοποιούμε την παράμετρο style ως εξής:

<h1 style="text-align: center;">Η γλώσσα CSS</h1>

Εδώ, στην επικεφαλίδα h1 έχει προστεθεί η παράμετρος style και συγκεκριμένα ορίζει την στοίχιση της να είναι στο κέντρο της σελίδας στο πρόγραμμα περιήγησης.

Ο δεύτερος τρόπος είναι η μορφοποίηση να γίνει σε ξεχωριστό αρχείο. Το αρχείο αυτό έχει κατάληξη .css και οι εντολές που περιλαμβάνει είναι ποικίλες. Η σύνδεση των δύο αρχείων γίνεται στο head tag με την προσθήκη της εντολής:

<link rel="stylesheet" type="text/css" href="style.css">

Με το αρχείο να έχει όνομα style.css (μπορεί να πάρει οποιοδήποτε αποδεκτό όνομα).

Είτε ο προγραμματιστής αποφασίσει να εντάξει το κομμάτι αυτό στο αρχείο html, είτε να το γράψει σε διαφορετικό αρχείο και να το συνδέσει μετά, οι εντολές που θα δούμε είναι ίδιες.

Πάμε να δούμε μερικές:

Color ⟶ Ορίζει το χρώμα ενός κειμένου, μια παραγράφου, μιας λέξεις, μιας κεφαλίδας ή ότι άλλο υπάρχει. Για παράδειγμα **p { color: blue; }**

Background-color ⟶ Ορίζει το χρώμα φόντου ενός στοιχείου, για παράδειγμα ολόκληρης της σελίδας ή ενός τμήματος ως **div { background-color: red; }**

Font-size ⟶ Αλλάζει το μέγεθος γραμματοσειράς ενός κειμένου. Για παράδειγμα **h1 { font-size: 24px; }** με το px να είναι η μονάδα μέτρησης.

Margin ⟶ Αν θέλουμε να δώσουμε ένα εξωτερικό περιθώριο γύρω από ένα στοιχείο, το margin δημιουργεί ανάμεσα αυτή την απόσταση και από τις τέσσερις μεριές (πάνω, κάτω, δεξιά, αριστερά). Για παράδειγμα: **div { margin: 10px; }**. Αν θέλουμε εσκεμμένα να δώσουμε την απόσταση σε μια μόνο από τις τέσσερις πλευρές, τότε η εντολή αλλάζει ως εξής: **margin-bottom: 20px;**

Η ένταξη τέτοιων μορφοποιήσεων στο html αρχείο μπορεί επίσης να πραγματοποιηθεί μέσω κλάσεων (**class**) μέσα σε ένα tag. Στην css οι κλάσεις ορίζονται με τελεία και ένα όνομα, πχ .result όπου συνοδεύετε με αγκύλες.

Παράδειγμα:
HTML

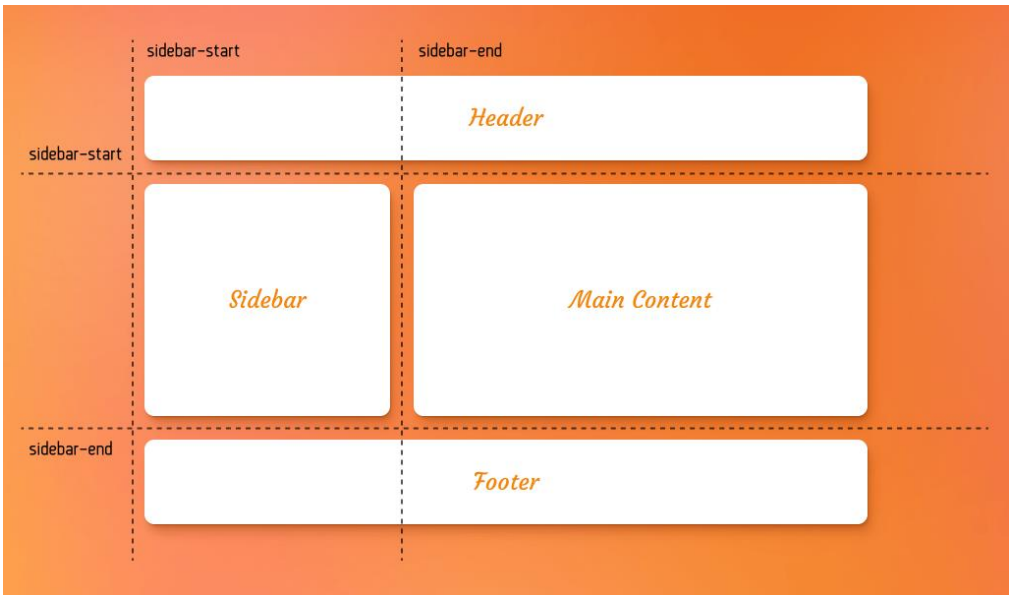
```
<div class=”result”>
    <p class=”magenta”>Ένα κείμενο με μορφοποίηση.</p>
</div>
```

CSS

```
. magenta {
   color: #e80fa3;
}

. result {
   font-size: 14px;
}
```

Ένα άλλο πολύ βασικό χαρακτηριστικό της γλώσσας CSS είναι ότι περιλαμβάνει συστήματα διάταξης, τα γνωστά **layouts**. Τα πιο γνωστά εργαλεία διάταξης είναι το flexbox κυρίως για την εμφάνιση φωτογραφιών και η διάταξη grid που περιέχει το βασικό πλέγμα που είναι δομημένες σχεδόν όλες οι ιστοσελίδες. Η διάταξη grid περιλαμβάνει διαφορετικά πεδία, τα οποία ουσιαστικά διαχωρίζουν την εφαρμογή σε τμήματα, με τα πιο δημοφιλή να είναι οι περιοχές: header, footer, nav, aside, main και section.



Η γλώσσα CSS κάνει μια εφαρμογή πιο όμορφη και ιδιαίτερη, ενώ μπορεί να τονίσει κάποια χαρακτηριστικά ενός κειμένου ή μιας φωτογραφίας στα οποία θέλει να δώσει έμφαση ο προγραμματιστής.

✓ JAVASCRIPT

Όπως αναφέρει και το βιβλίο: «Τεχνολογίες και Προγραμματισμός στον Παγκόσμιο Ιστό», η **JavaScript** είναι μια γλώσσα σεναρίων και εξυπηρετεί τον προγραμματισμό εφαρμογών που εκτελούνται από τον πελάτη. Εμφανίστηκε το 1995 από τον Brendan Eich και σχεδιάστηκε για να τρέχει στατικές ιστοσελίδες Html στον browser. Σήμερα όμως, έχει εξελιχθεί σε μια από τις δημοφιλέστερες γλώσσες προγραμματισμού, με διαφορετικές λειτουργίες και δυνατότητες για τον προγραμματιστή που πλέον υπάρχουν εργαλεία που φτιάχτηκαν μέσω αυτής της γλώσσας για χρήση τόσο στο frontend, όσο και στο backend. Η γλώσσα αυτή διαχειρίζεται συναρτήσεις, event και λειτουργίες στέλνοντας δεδομένα και πληροφορίες για μια ενέργεια που έχει κάνει ο χρήστης στην ιστοσελίδα. Η ενέργεια του μπορεί να έχει να κάνει για παράδειγμα με το πάτημα ενός κουμπιού όπου αυτό το κουμπί (button) θα χρειαστεί να εκτελέσει μια άλλη ενέργεια όπως την εμφάνιση ενός μενού, την εμφάνιση ενός κειμένου, την μεταφόρτωση του χρήστη σε κάποια άλλη ιστοσελίδα και πολλά ακόμη. Συγκεκριμένα, το βιβλίο αναφέρει ότι «η JavaScript είναι μια γλώσσα σεναρίων που εκτελείται από την πλευρά του πελάτη και η οποία κάνει τις ιστοσελίδες πιο δυναμικές και διαδραστικές. Ως γλώσσα σεναρίων, είναι μια σχετικά απλή γλώσσα προγραμματισμού. Η JavaScript επιτρέπει τη συγγραφή μικρών προγραμμάτων, γνωστών ως σενάρια, τα οποία φορτώνονται και εκτελούνται στον φυλλομετρητή του χρήστη. Παραδείγματος χάριν, ένα πρόγραμμα το οποίο ελέγχει εάν έχουν συμπληρωθεί σωστά τα πεδία μιας φόρμας και ειδοποιεί τον χρήστη για τυχόν σφάλματα, προτού τα στοιχεία σταλούν στον εξυπηρετητή.»

Ο κώδικας γραμμένος στην γλώσσα αυτή συνδέεται όπως και το αρχείο .css με το αρχικό μας αρχείο html. Υπάρχουν και εδώ δύο τρόποι ενσωμάτωσης.

Ο πρώτος τρόπος πραγματοποιείται με την προσθήκη του tag **<script> … </script>** και μέσα εκεί γράφεται ο κώδικας. Σε διαφορετική περίπτωση μπορούμε να τον τοποθετήσουμε σε ένα τελείως ξεχωριστό αρχείο που θα έχει την κατάληξη .js και να το συνδέσουμε στο αρχείο html, βάζοντας στο head του κώδικα την εντολή:
<script src="script.js"></script>
Με το αρχείο κώδικα σε γλώσσα JavaScript να είναι το script.js .

Πάμε να δούμε πέντε βασικές εντολές γραμμένες στην γλώσσα.

let – const ⟶ Χρησιμοποιούνται για την δημιουργία μεταβλητών και σταθερών ως: **let age = 23;** Και **const gender = “female”;**

console.log() ⟶ Εκτυπώνει ένα μήνυμα, μια μεταβλητή, κάποια σταθερά στην κονσόλα. Για παράδειγμα: **console.log(“Hello!”);**

function ⟶ Ορίζει μια συνάρτηση για να μπορεί αν θέλει ο προγραμματιστής να την καλεί επανειλημμένα.

document.getElementById ⟶ Αυτή η εντολή χρησιμοποιείται για την επιλογή και τον χειρισμό στοιχείων, τα οποία καλούνται DOM (Document Object Model). Το document είναι ένα αντικείμενο. Το getElementById είναι μια μέθοδος ή αλλιώς συνάρτηση αναζήτησης ενός html στοιχείου βάση το μοναδικό του id. Για παράδειγμα: **document.getElementById('cVar').onclick = function() {…}** . Εδώ παίρνουμε το στοιχείο με id cVar και καλούμε μια άλλη συνάρτηση της JavaScript με όνομα onclick, που σημαίνει ότι το στοιχείο cVar μπορεί να «πατηθεί» από τον χρήστη στην ιστοσελίδα πχ σαν κουμπί και όταν ο χρήστης το κάνει θα εκτελεστούν οι εντολές που

υπάρχουν μέσα στις αγκύλες της συνάρτησης. Το id στο αρχείο html μπορεί να αντιπροσωπεύεται κάπως έτσι: **<button class="button" , id="cVar">📌 Μεταβλητές και Σταθερές</button>** .

for ⟶ Είναι η κλασική εντολή επανάληψης που εκτελεί μια ενέργεια για προκαθορισμένο αριθμό επαναλήψεων. Η JavaScript υποστηρίζει και άλλες δομές, όπως η δομή επιλογής **if**.

Οι δυνατότητες της γλώσσας είναι άπειρες και απολύτως χρήσιμες για μια σύγχρονη εφαρμογή. Η JavaScript υποστηρίζει την αντικειμενοστραφή προγραμματιστική λογική, δίνοντας τη δυνατότητα δημιουργίας και διαχείρισης αντικειμένων, κλάσεων και μεθόδων. Εξυπηρετεί δυναμικά σάιτ στον φυλλομετρητή του χρήστη, προσθέτοντας διαδραστικότητα όπως η διαχείριση event και καθιστά την γλώσσα ως μια που βασίζεται στην πλευρά του χρήστη και γενικότερα στο frontend. Έχει παρόλα αυτά την δυνατότητα να χρησιμοποιηθεί και στην πλευρά του διακομιστή και με λίγα λόγια στο backend. Αυτό έγινε με την έλευση του **Node.js** ενός εργαλείου διαχείρισης server, για το οποίο θα μιλήσουμε αργότερα.

Ένα απλό παράδειγμα κώδικα της γλώσσας, αποτελείται από μια ιστοσελίδα με ένα κουμπί που όταν το πατήσει ο χρήστης, αυτό εμφανίζει την ημερομηνία και την ώρα με την βοήθεια μιας μεθόδου είναι το εξής:

```
<!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>
```

3.1.β: ΤΕΧΝΟΛΟΓΙΕΣ BACKEND

Οι σύγχρονες ιστοσελίδες που κυκλοφορούν στο διαδίκτυο στην πλειονότητα τους είναι δυναμικές εφαρμογές. Πλατφόρμες έχουν για παράδειγμα την δυνατότητα σύνδεσης και αποσύνδεσης ενός χρήστη, την αποθήκευση δεδομένων, την προσαρμογή της ιστοσελίδας πάνω σε έναν συγκεκριμένο χρήστη κ.α. Όλη αυτή η μαγεία της αρχιτεκτονικής του σάιτ ή ενός app και του τρόπου λειτουργίας του γίνεται με τεχνολογίες **backend**.

Τεχνολογίες σαν αυτές αναπτύσσουν τις εφαρμογές όχι πλέον από την οπτική του χρήστη όπως οι front end, αλλά από την μεριά του διακομιστή γνωστού ως server. Στην κατηγορία αυτή ανήκουν γλώσσες προγραμματισμού όπως **Python, PHP, Java, JavaScript** ενώ συμπεριλαμβάνονται επίσης

και συστήματα αποθήκευσης δεδομένων όπως βάσεις δεδομένων. Οι βάσεις συχνά εξυπηρετούνται με την γλώσσα προγραμματισμού SQL, με εργαλεία να είναι τα MySQL ή MongoDB. Ουσιαστικά ένας backend developer σχεδιάζει APIs με τα οποία μπορεί η εφαρμογή να στέλνει και να λαμβάνει δεδομένα από το frontend στο backend, κάνοντας τα δύο τμήματα της εφαρμογής να επικοινωνούν.

Αναφέραμε ότι το backend ασχολείται με την ανάπτυξη κώδικα από την πλευρά του server. Αυτό σημαίνει ότι τα δεδομένα με τα οποία ασχολείται ο προγραμματιστής δεν θα είναι ορατά στον χρήστη, αλλά επιτρέπει να γίνονται ενέργειες από την μεριά του χρήστη με σκοπό την παραγωγή μιας ενέργειας. Για παράδειγμα αν υπάρχει στην ιστοσελίδα ένα κουμπί (button) για σύνδεση στην εφαρμογή, ο χρήστης όταν το πατήσει λογικό είναι να περιμένει να συμβεί κάτι. Το αποτέλεσμα της ενέργειας επεξεργάζεται από τον server και δείχνει στον πελάτη το αντίστοιχο αποτέλεσμα.

Ας δούμε μερικές τεχνολογίες back end!

- **PHP**

Η **PHP** είναι μια δημοφιλής γλώσσα προγραμματισμού για ανάπτυξη δυναμικών ιστοσελίδων. Αναπτύχθηκε την δεκαετία του ’90 και τα αρχικά της προέρχονται από το Personal Home Page, όμως η ερμηνεία τους έχει αλλάξει και πλέον σημαίνει Hypertext Preprocessor. Είναι σχετικά απλή για να την μάθει ένας προγραμματιστής και υποστηρίζει πολλές δυνατότητες.

Εκτελείται ως μια server side γλώσσα προγραμματισμού, πράγμα που σημαίνει ότι επεξεργάζεται από τον web server και το τελικό αποτέλεσμα αποστέλλεται στον φυλλομετρητή του χρήστη.

Όταν λέμε ότι η γλώσσα χρησιμοποιείται για ανάπτυξη δυναμικών ιστοσελίδων, εννοούμε ότι η ιστοσελίδα έχει την ευκαιρία να προσαρμόζεται δυναμικά σε έναν χρήστη και να αλλάζει τα δεδομένα της για παράδειγμα βάσει μιας καταχώρησης δεδομένων. Συχνά η php συνδυάζεται με κάποια **βάση δεδομένων** για εύκολη αποθήκευση, ανάκτηση, διαγραφή και επεξεργασία δεδομένων σε web εφαρμογές. Διαθέτει επίσης μια μεγάλη ποικιλία από εργαλεία και βιβλιοθήκες, κάνοντας πιο απλή την δουλειά μας ως προγραμματιστές.

Τα εργαλεία που χρησιμοποιεί για την αλληλεπίδραση της με τις βάσεις δεδομένων είναι το MySQL και το PostgreSQL. Η PHP είναι ιδιαίτερα χρήσιμη για την επεξεργασία δεδομένων από φόρμες HTML, όπως login, forms κ.α. Επεξεργάζεται τα δεδομένα που εισάγουν οι χρήστες και μπορεί να τα χρησιμοποιήσει για διάφορες ενέργειες, όπως η αποθήκευση σε βάση δεδομένων ή η αποστολή email.

- ✓ Η γραφή του κώδικα της γλώσσας γίνεται ανάμεσα από τα σύμβολα (tags):
  <?php … ?>
- ✓ Η αποστολή δεδομένων από ένα html αρχείο όπως τα προηγούμενα γίνεται με τις μεθόδους POST και GET.

Η **GET** χρησιμοποιείται για ανάκτηση δεδομένων από τον server και υπάρχει περιορισμός στο μέγεθος τους. Η σύνταξη της ορίζεται ως:

```
<form action="search.php" method="GET">
    Όνομα: <input type="text" name="name">
    <input type="submit" value="Αναζήτηση">
</form>
```

Η μέθοδος **POST** από την άλλη, χρησιμοποιείται για αποστολή δεδομένων στον server. Είναι κατάλληλη για αποστολή μεγαλύτερων δεδομένων όπως κωδικοί πρόσβασης και την προτιμούμε για αποθήκευση τους σε κάποια βάση.

```
<form action="send.php" method="POST">
    Όνομα: <input type="text" name="name">
    <input type="submit" value="Υποβολή">
</form>
```

Είναι δύο πολύ γνωστές μέθοδοι επεξεργασίας δεδομένων σε server που τις χρησιμοποιούν και άλλες γλώσσες προγραμματισμού.

Παρά τα οφέλη που έχει αυτή η γλώσσα και είναι αποδοτική και γρήγορη για πολλές εφαρμογές, υπάρχουν άλλες λύσεις για εξαιρετικά απαιτητικά project όπως ένα εργαλείο φτιαγμένο σε JavaScript το Node.js που θα αναλύσουμε παρακάτω. Βέβαια μέχρι και τα τελευταία χρόνια έχουν κυκλοφορήσει διάφορες εκδόσεις της γλώσσας με βελτιώσεις σε ήδη υπάρχοντα προβλήματα, νέες κυκλοφορίες και μεγαλύτερη ασφάλεια. Η νεότερη έκδοση της βγήκε στην αγορά το 2020 και αναφερόμαστε στην PHP 8, η οποία έχει υποστεί επιπλέον αλλαγές με την νεότερη την 8.3 που βγήκε το 2023.

Ένα παράδειγμα σε γλώσσα PHP:

```
<!DOCTYPE html>
<html>

  <head>
    <title>PHP "Hello, World!" program</title>
  </head>

  <body>
    <p> <?= 'Hello, World!' ?> </p>
  </body>

</html>
```

Υπάρχει τρόπος να προσθέσουμε τον κώδικα γραμμένο σε PHP και σε ξεχωριστό αρχείο με την κατάληξη **.php** . Για παράδειγμα:

```
<?php
echo 'Hello, everyone!';
?>
```

Με τα δεδομένα να στέλνονται στον server και να τυπώνονται στην οθόνη του χρήστη.

Μια βασική απλή εντολή της γλώσσας είναι αυτή που υπάρχει στο προηγούμενο παράδειγμα κώδικα, η εντολή **echo**. Η εντολή μπορεί να είναι γνωστή σε μερικούς προγραμματιστές, αφού χρησιμοποιείται συνήθως στην ανάπτυξη κώδικα σε scripts. Αποτελεί την εντολή που χρησιμοποιείται για την εκτύπωση ενός περιεχομένου (κειμένου, μεταβλητών κ.α.) στην οθόνη.

Η γλώσσα επιπλέον υποστηρίζει δομές επιλογής όπως **if**, if…else καθώς και δομές επανάληψης όπως **for, while** με παρόμοιο τρόπο με αυτόν που αναλύσαμε νωρίτερα στην εργασία για την γλώσσα προγραμματισμού C. Επιπρόσθετα επιτρέπεται η χρήση **συναρτήσεων** και **μεταβλητών**, όπου επίσης οι **τύποι δεδομένων** (integer, float, string, Boolean κ.α.) είναι παρόμοιοι με αυτούς της γλώσσας C και δέχεται επίσης την επεξεργασία **αρχείων**.

```
<?php
function hi($name) {
   echo "Hello, $name!";
}

hi("everyone");
?>
```

Η έξοδος του κώδικα θα είναι η ίδια με το προηγούμενο παράδειγμα απλώς με χρήση συνάρτησης.

Μια ευρέως γνωστή εφαρμογή που χρησιμοποιεί την γλώσσα PHP είναι το Facebook. Η εταιρεία συνδυάζει την τεχνολογία αυτή με την βάση δεδομένων MySQL.

➢ Python

Η γλώσσα προγραμματισμού **python** αποτελεί μια απλοϊκή και χωρίς πολλούς περιορισμούς γλώσσα με δυνατότητες ανάπτυξης προγραμμάτων, από απλά project μέχρι και εκπαίδευση μοντέλων τεχνητής νοημοσύνης. Στον τομέα του back end χρησιμοποιείται συνήθως με κάποια frameworks όπως το Django ή το Flask. Τα αρχεία σε γλώσσα python έχουν κατάληξη **.py** και εκτελούνται από το terminal σε μια καθορισμένη θύρα (port) συνήθως την 5000 για το flask.

Το **Django** είναι ένα κορυφαίο framework, με ενσωματωμένα συστήματα για αυθεντικοποίηση, ORM (Object Relational Mapping) και άλλα απαραίτητα στοιχεία τα οποία προσφέρουν μια ολοκληρωμένη προσέγγιση για την ανάπτυξη εφαρμογών.

Το **Flask** είναι ένα πιο «ελαφρύ» εργαλείο με περισσότερη ευελιξία. Επιτρέπει την επιλογή στοιχείων ανάλογα με τις απαιτήσεις του προγράμματος χωρίς να επιβάλλει ένα συγκεκριμένο σύνολο εργαλείων όπως το Django. Τα στοιχεία μπορεί να είναι για παράδειγμα βιβλιοθήκες που θέλει να ενσωματώσει στο πρόγραμμα του ο προγραμματιστής. Είναι ιδανικό για μικρότερα projects.

Για παράδειγμα:

Εισάγουμε την βιβλιοθήκη flask (from flask import Flask). Δημιουργούμε ένα αντικείμενο της κλάσης Flask και ορίζουμε μια διαδρομή (έστω /) για να εκτυπωθεί το μήνυμα (Hello from Flask) όταν επισκεπτόμαστε την σελίδα της εφαρμογής. Τέλος, κάνουμε εκκίνηση των server και βλέπουμε αν υπάρχουν σφάλματα (debugging).

```
from flask import Flask

app = Flask(__name__)

@app.route('/')
def home():
    return "Hello from Flask!"

if __name__ == '__main__':
    app.run(debug=True)
```

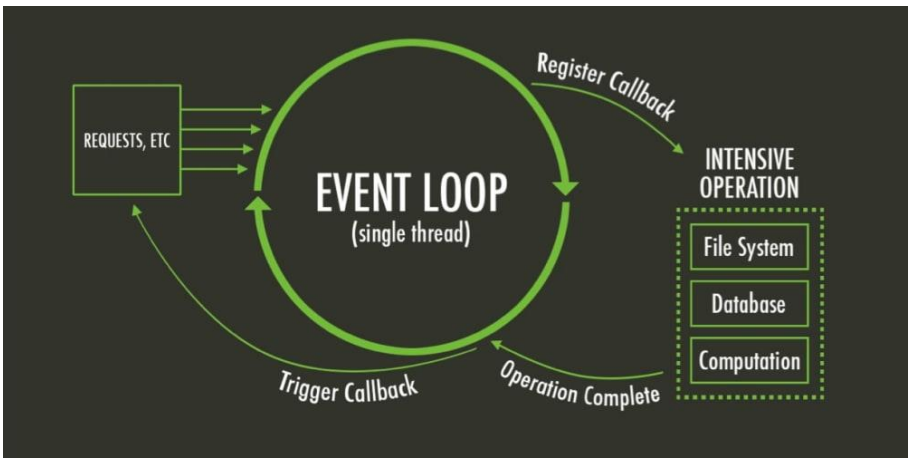
Η Python είναι κατάλληλη για ανάπτυξη APIs, μεγάλες και πολύπλοκες εφαρμογές web, καθώς και για εφαρμογές με βάσεις δεδομένων.

- **Node.js**

Το **Node.js** πρόκειται για ένα εργαλείο φτιαγμένο σε γλώσσα JavaScript, λειτουργώντας από την μεριά του server και επιτρέποντας την εκτέλεση κώδικα στον διακομιστή. Δημιουργήθηκε το 2009 για ανάπτυξη backend υπηρεσιών, όπως APIs και εφαρμογές που εκτελούνται σε πραγματικό χρόνο (real-time) όπως για παράδειγμα οι συνομιλίες. Είναι μια τεχνολογία open source και βασίζεται στην μηχανή V8 της Google.

Είναι μια πολύ δημοφιλής μέθοδος λόγω της δυνατότητας του να χρησιμοποιεί κανείς την JavaScript και ως frontend και ως backend τεχνολογία ανάπτυξης ιστοσελίδων. Επιτρέπει, δηλαδή, στους προγραμματιστές να εκτελούν το κώδικα απευθείας στον **server** εκτός του φυλλομετρητή. Παράλληλα λειτουργεί ως ένα non-blocking thread.

Βασίζεται σε διαχείριση **event** και μπορεί να χειρίζεται ταυτόχρονα πολλά αιτήματα για αποστολή ή λήψη δεδομένων στον server, χωρίς πολλές καθυστερήσεις. Δεν χρειάζεται λοιπόν, να περιμένουμε να ολοκληρωθεί μια διαδικασία για να ξεκινήσει μια άλλη. Μπορεί επίσης να χρησιμοποιηθεί σε εφαρμογές με μεγάλη απαιτητικότητα όπως πλατφόρμες streaming. Επιπλέον, όπως και οι προηγούμενες τεχνολογίες που είδαμε, έτσι και το node.js μπορεί να συνδεθεί με συστήματα βάσεων δεδομένων για μόνιμη αποθήκευση πληροφοριών όπως το MySQL.



Το αρχείο διαχείρισης του server έχει και αυτό την ίδια κατάληξη με τα JavaScript αρχεία κώδικα, το **.js** .

Πάμε να δούμε μερικές βασικές εντολές!

require() ⟶ Χρησιμοποιείται για την εισαγωγή βιβλιοθηκών ή ενσωματωμένων modules του node.js σε ένα αρχείο. Για παράδειγμα το module http εισάγεται ως: const http = require('http');

console.log() ⟶ Χρησιμοποιείται για την εκτύπωση μηνυμάτων στην κονσόλα της εφαρμογής. Για παράδειγμα: console.log('Hello!');

http.createServer() ⟶ Δημιουργεί έναν νέο http server για την διαχείριση αιτημάτων (requests) και απαντήσεων (responses). Για παράδειγμα:

```
const server = http.createServer(function (req, res) {

    res.writeHead(200, {'Content-Type' : 'text/html'});
    res.end('Hello from Node.js');
});
```

server.listen() ⟶ Είναι μια μέθοδος που χρησιμοποιείται για να «ακούει» ο server σε μια συγκεκριμένη θύρα (port) που του έχουμε ορίσει εμείς. Για παράδειγμα:

```
const port = 4000;
server.listen(port, function(){
    console.log("server listening at " + port)
})
```

fs.readFile() ⟶ Το node.js υποστηρίζει την ανάγνωση και επεξεργασία αρχείων μέσω του module fs (file system). Αυτή η εντολή είναι υπεύθυνη για την ανάγνωση αρχείων. Για παράδειγμα:

```
const fs = require('fs');

fs.readFile ('example.txt', 'utf8', (err, data) => {
    if (err) throw err;
        console.log(data);
});
```

Αναφέραμε επίσης, ότι ο server αυτός λειτουργεί με **events** τα οποία είναι βασική αρχή για την αρχιτεκτονική του. Στο μοντέλο βασισμένο σε events ενέργειες ή λειτουργίες εκπέμπουν (emit) γεγονότα (events) που μπορούν να εκτελούνται από χειριστές (event handler). Για να επιτευχθεί αυτό υπάρχουν δύο έτοιμες μέθοδοι για να μας βοηθήσουν. Η **emit** χρησιμοποιείται για να «ενεργοποιήσει» ένα event και η **on** για προσθέσει μια συνάρτηση επανάκλησης που πρόκειται να εκτελεστεί όταν ενεργοποιηθεί το event. Πάμε να δούμε ένα παράδειγμα:

```
const EventEmitter = require('events');
const eventEmitter = new EventEmitter();

eventEmitter.on('start', () => {
    console.log('started');
});

eventEmitter.emit('start');
```

Ένα ακόμη πλεονέκτημα του node.js είναι το πακέτο **NPM**. Τα αρχικά του προέρχονται από το node package manager και αποθηκεύονται στην συσκευή μας κατά την διάρκεια εγκατάστασης του node.js εργαλείου. Αποτελεί το μεγαλύτερο περιβάλλον ανοιχτού κώδικα πακέτων και προσφέρει ουσιαστικά μια ακόμα μεγαλύτερη γκάμα έτοιμων πακέτων όπως βιβλιοθήκες, για να μπορέσουμε ως προγραμματιστές να επεκτείνουμε εύκολα την λειτουργικότητα των εφαρμογών μας. Για παράδειγμα: **npm install upper-case**

Το εργαλείο Node.js εμπεριέχει επίσης και άλλα εργαλεία που είναι πολύ βοηθητικά για την δημιουργία σύγχρονων δυναμικών ιστοσελίδων όπως αυτή που θέλουμε να φτιάξουμε στην εργασία μας. Ο λόγος γίνεται για δύο βασικές υπηρεσίες, τα εργαλεία **Express** και **AJAX**. Αυτά, συχνά συνδυάζονται με έναν node.js server για την κατασκευή διαδραστικών και δυναμικών web εφαρμογών, με το Express να χειρίζεται τον server side κώδικα και το AJAX να διαχειρίζεται τις ασύγχρονες αιτήσεις από το frontend προς το backend.

Το express είναι ένα framework για το node.js που διευκολύνει την δημιουργία web server και **API endpoints** (για παράδειγμα /data). Κύρια λειτουργία του αποτελεί η δημιουργία διαδρομών, γνωστά ως routes και ο χειρισμός αιτημάτων όπως οι δύο μέθοδοι **POST** και **GET**, για τις οποίες μιλήσαμε σε προηγούμενη ενότητα. Δημιουργεί τη δομή του server που χειρίζεται τα αιτήματα και απαντά με δεδομένα.

Το AJAX (Asynchronous JavaScript and XML) είναι μια τεχνική που επιτρέπει στον χρήστη να στέλνει δεδομένα από το frontend και να λαμβάνει δεδομένα από τον server (από το backend) ασύγχρονα, χωρίς να απαιτείται από τον χρήστη να ανανεώσει την σελίδα στον browser. Συνδέεται με τα node.js και express για να στέλνει αιτήματα στον server και να ενημερώνει δυναμικά το περιεχόμενο της ιστοσελίδας. Η αποστολή ενός αιτήματος και εδώ επιτυγχάνεται μέσω των μεθόδων **GET** και **POST**. Στον server, δημιουργούμε ένα API endpoint, το οποίο δέχεται αιτήματα και επιστρέφει μια απάντηση. Στη συνέχεια, το frontend χρησιμοποιεί AJAX για να λάβει τα δεδομένα.

Περισσότερες πληροφορίες για τα εργαλεία αυτά θα συναντήσουμε και στα παρακάτω κεφάλαια.

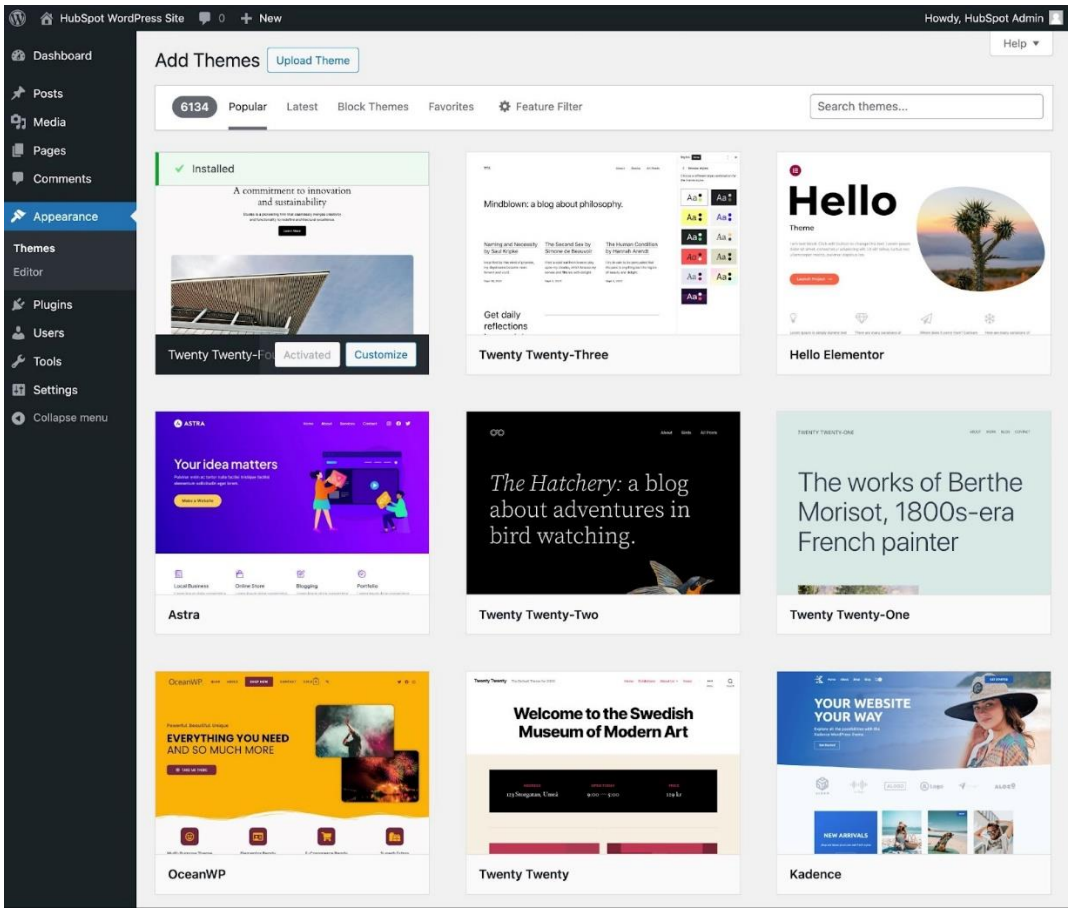
3.1.γ: ΆΛΛΑ ΕΡΓΑΛΕΙΑ ΕΦΑΡΜΟΓΩΝ

Οι δύο μεγάλες προηγούμενες τεχνολογίες frontend και backend περιέχουν τα βασικά εργαλεία ανάπτυξης εφαρμογών και κατά συνέπεια εφαρμογές εκμάθησης γλωσσών προγραμματισμού. Υπάρχουν όμως και έτοιμα εργαλεία δημιουργίας ιστοσελίδων που συνδυάζουν τις παραπάνω τεχνολογίες σε ένα κοινό περιβάλλον. Ένα τέτοιο παράδειγμα είναι το εργαλείο **WordPress**.

- **WordPress**

Το WordPress θεωρείται μια τεχνολογία διαχείρισης περιεχομένου. Είναι χτισμένο κυρίως πάνω στην γλώσσα προγραμματισμού PHP και χρησιμοποιεί την MySQL ως βάση δεδομένων για αποθήκευση του περιεχομένου, των ρυθμίσεων, των χρηστών και άλλων δεδομένων της ιστοσελίδας. Στο frontend χρησιμοποιεί html για την δομή των εφαρμογών, CSS για την μορφοποίηση και την εμφάνιση τους και JavaScript για την προσθήκη διαδραστικών δυνατοτήτων. Αρχικά κυκλοφόρησε το 2003 και έχει πλέον εξελιχθεί σε ένα ισχυρό εργαλείο που χρησιμοποιούν εκατομμύρια προγραμματιστές και επιχειρήσεις, καθώς αποτελεί και αυτό ένα open source λογισμικό, το οποίο σημαίνει ότι είναι ελεύθερος για χρήση από οποιονδήποτε, όπως επίσης σημαίνει ότι μπορεί να αλλάξει και να τροποποιηθεί από τον προγραμματιστή.

Ένα ακόμη βασικό πλεονέκτημα του είναι η ευκολία χρήσης του. Ακόμα και χρήστες που δεν είναι έμπειροι προγραμματιστές ή δεν έχουν τεχνικές γνώσεις μπορούν να δημιουργήσουν και να διαχειριστούν τις ιστοσελίδες τους. Προσφέρει ένα τεράστιο εύρος από θέματα – **themes**, με διαφορετικά σχέδια και στυλ για να επιλέξει κάποιος για την εφαρμογή του, κάτι που το κάνει ακόμα πιο βατό για έναν χρήστη να προσαρμόσει την εμφάνιση της ιστοσελίδας του.



Συνολικά, το WordPress συνδυάζει τεχνολογίες backend (PHP, MySQL) και frontend (HTML, CSS, JavaScript), προσφέροντας μια ολοκληρωμένη λύση για τη δημιουργία και τη διαχείριση ιστοσελίδων. Είναι ιδιαίτερα δημοφιλές λόγω της ευελιξίας του, της μεγάλης κοινότητας υποστήριξης και των συνεχών αναβαθμίσεων που βελτιώνουν την ασφάλεια και την απόδοση.

➢ Βάσεις Δεδομένων

Όπως έχουμε αναφέρει κατά διαστήματα σε μερικές από τις ενότητες τις εργασίας, ένα ακόμη πολύ βασικό εργαλείο για τις σύγχρονες και δυναμικές ιστοσελίδες είναι οι **βάσεις δεδομένων – data bases**. Μια ιστοσελίδα δεν μπορεί να διατηρήσει τα δεδομένα και της πληροφορίες της αν είναι δυναμική, διότι αλλάζει συνεχώς. Για την μόνιμη, λοιπόν, αποθήκευση δεδομένων όπως οι χρήστες της, οι πρόοδος των χρηστών, το email τους και διάφορες άλλες πληροφορίες έχουν δημιουργηθεί τα συστήματα βάσεων δεδομένων.

Οι βάσεις δεδομένων είναι συστήματα που χρησιμοποιούνται για την αποθήκευση, οργάνωση και διαχείριση δεδομένων με τρόπο που επιτρέπει την εύκολη πρόσβαση, τροποποίηση και ανάκτηση αυτών των δεδομένων. Οι βάσεις δεδομένων παίζουν καθοριστικό ρόλο στις εφαρμογές, ειδικά στις διαδικτυακές εφαρμογές, όπου πρέπει να αποθηκεύονται δεδομένα χρηστών, προϊόντα, παραγγελίες και άλλες πληροφορίες.

Υπάρχουν δύο βασικές κατηγορίες βάσεων δεδομένων, οι λεγόμενες **relational databases** και οι **no SQL databases**.

❖ Relational Databases (RDB):

Οι βάσεις αυτές οργανώνουν τα δεδομένα τους σε πίνακες (tables) που μοιάζουν με τους κλασικούς πίνακες προγραμματισμού που γνωρίζουμε, δηλαδή αποτελούνται από σειρές (rows) και από στήλες (columns). Κάθε πίνακας έχει μια συγκεκριμένη κατηγορία δεδομένων και μπορούν να συνδέονται μεταξύ τους με σχέσεις μέσω συνήθως των κοινών τους πεδίων που λέγονται πρωτεύοντα κλειδιά (primary keys).

- Η γλώσσα στην οποία γράφονται τα δεδομένα είναι η **SQL** – Structured Query Language.
- Μερικά συστήματα που μπορείς να δημιουργήσεις βάσεις δεδομένων γραμμένες στην γλώσσα αυτή είναι τα **MySQL** και **PostgreSQL**.
- Ένα παράδειγμα κώδικα δημιουργίας ενός πίνακα με χρήστες μπορεί να είναι:

```
CREATE TABLE Customers  (
    id INT PRIMARY KEY,
    name VARCHAR(50),
    email VARCHAR(100)
);

INSERT INTO Customers  (id, name, email)
VALUES (1, 'John Doe', 'john@example.com');

SELECT * FROM Customers ;
```

❖ No SQL Databases:

Αν και ο όρος των μη σχεσιακών βάσεων δεδομένων, δηλαδή των βάσεων που δεν εξυπηρετούνται με την γλώσσα SQL ακούστηκε για πρώτη φορά το 1998 από τον Carlo Strozzi, στο φως της αγοράς ήρθε το 2009. Τότε ο Eric Evans μίλησε και είπε ότι: « the whole point of seeking alternatives is that you need to solve a problem that relational databases are a bad fit for».

Η πρόκληση που είχαν να αντιμετωπίσουν οι RDB είχε να κάνει με την αποθήκευση και επεξεργασία μεγάλου όγκου δεδομένων που απαιτούν υψηλή διαθεσιμότητα. Αυτό που εννοεί με την "υψηλή διαθεσιμότητα" είναι ότι τέτοιες εφαρμογές πρέπει να είναι προσβάσιμες και λειτουργικές, χωρίς διακοπές. Οι σχεσιακές βάσεις δεδομένων (RDB) αντιμετώπισαν προκλήσεις με αυτό το μοντέλο, καθώς δεν είχαν σχεδιαστεί για να διαχειρίζονται τεράστια ποσά δεδομένων ή να επεκτείνονται (scalability) με την ίδια ευκολία. Για αυτό τον λόγο, πολλές εταιρείες άρχισαν να υιοθετούν No SQL βάσεις δεδομένων.

Οι **No SQL** βάσεις δεδομένων είναι κατάλληλες για εφαρμογές με μεγάλα ή πολύπλοκα δεδομένα που δεν χρειάζονται αυστηρές σχέσεις. Σήμερα υπάρχουν πάνω από 150 No SQL προϊόντα με διαφορετικές χρήσεις το καθένα.

Μερικά παραδείγματα είναι τα **MongoDB**, **Redis** και **Cassandra**.

```
{
   "_id": 1,
   "name": "John Doe",
   "email": "john@example.com"
}
```

Βεβαίως υπάρχουν και άλλες βασικές διαφορές ανάμεσα στις δύο κατηγορίες τις οποίες ένας προγραμματιστής πρέπει να περιεργαστεί, για να μπορέσει να αποφασίσει πιο εργαλείο είναι το πιο κατάλληλο στο δικό του project.

ΚΕΦΑΛΑΙΟ 4 ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΕΦΑΡΜΟΓΗΣ

Στο προηγούμενο κεφάλαιο είδαμε πολλές τεχνολογίες ανάπτυξης εφαρμογών. Σε αυτό το κεφάλαιο θα μιλήσουμε πιο συγκεκριμένα για τις εφαρμογές που χρησιμοποιήθηκαν για την ανάπτυξη της διαδικτυακής μας πλατφόρμας.

4.1: ΓΕΝΙΚΟ ΠΛΑΙΣΙΟ ΕΦΑΡΜΟΓΗΣ

Οι εφαρμογές εκμάθησης γλωσσών προγραμματισμού μπορούν να υλοποιηθούν με διαφορετικούς τρόπους, κάποιοι από τους οποίους αναφέρθηκαν στο προηγούμενο κεφάλαιο. Όσο αφορά τις τεχνολογίες front end, οι επιλογές που έχουμε ως προγραμματιστές δεν είναι τόσες πολλές. Από την άλλη οι τεχνολογίες back end έχουν πληθώρα επιλογών.

Για την καλύτερη κατανόηση της σημασίας εκμάθησης γλωσσών προγραμματισμού μέσω μιας ιστοσελίδας, αναπτύξαμε μια πλατφόρμα για την διδασκαλία και την μάθηση των χρηστών βασισμένη στην γλώσσα C. Αποφασίσαμε να μην προχωρήσουμε σε ανάπτυξη της μέσω έτοιμων τεχνολογιών διαχείρισης περιεχομένου -όπως το WordPress- και να ακολουθήσουμε σε πιο «παραδοσιακές» μεθόδους που συνδυάζουν τεχνολογίες **front end, back end και βάσεις δεδομένων**.

Οι τεχνολογίες αυτές όταν συνδυαστούν αποδοτικά και σωστά μπορούν να μας χαρίσουν ένα όμορφο περιβάλλον για τον προγραμματισμό της εφαρμογής. Επιλέχθηκαν, μετά από έρευνα, οι γλώσσες: **HTML, CSS, JavaScript, Node.js και PostgreSQL**. Με αυτά τα μέσα λύθηκαν βασικά προβλήματα που αντιμετωπίσαμε κατά την διαδικασία συγγραφής της διαδικτυακής εφαρμογής και για κάθε εργαλείο επιλέχθηκαν συγκεκριμένες ενέργειες και κατάλληλες αρμοδιότητες για την επιτυχή ολοκλήρωση της.

Ο τρόπος με τον οποίο λειτουργεί η εφαρμογή θα αναλυθεί σε επόμενο κεφάλαιο. Αξίζει, όμως, να επισημάνουμε πως ο χρήστης για να έχει πρόσβαση στα κεφάλαια των μαθημάτων της εφαρμογής πρέπει πρώτα να εγγραφεί στην πλατφόρμα και έπειτα να συνδεθεί. Έπειτα, ο χρήστης έχει πρόσβαση στην θεωρία, στις ασκήσεις, σε τεστ, σε παραδείγματα κώδικα, σε βαθμολογίες και σε πολλές ακόμη δυνατότητες που προσφέρει η ιστοσελίδα. Τα δεδομένα του αποθηκεύονται σε μια βάση δεδομένων για να μπορεί να γίνει επαλήθευση των στοιχείων του κάθε φορά που συνδέεται, ακόμα και όταν ο server της εφαρμογής κλείνει. Ο μαθητής ή ο νέος προγραμματιστής έχει πρόσβαση σε όλα τα κεφάλαια της γλώσσας C ανεξάρτητα του επιπέδου μάθησης του.

Ως επί το πλείστων η πλατφόρμα που δημιουργήθηκε ανήκει στην κατηγορία των διαδικτυακών εφαρμογών και πιο συγκεκριμένα στην κατηγορία που δεν φέρουν κάποια πιστοποίηση μετά το πέρας των μαθημάτων. Περισσότερες πληροφορίες για αυτού του είδους τις εφαρμογές υπάρχουν στο δεύτερο κεφάλαιο.

Παρακάτω θα δούμε ένα προς ένα τα εργαλεία που χρησιμοποιήθηκαν, δίνοντας έμφαση στον τρόπο που τα χειριστήκαμε στην εφαρμογή.

4.2: HTML

Στην εφαρμογή υπάρχουν συνολικά επτά αρχεία γραμμένα σε γλώσσα HTML. Λόγω της ύπαρξης ενός server τρία από τα επτά αρχεία, το βασικό μέρος της εφαρμογής, η σύνδεση του χρήστη και η εγγραφή του στην εφαρμογή, έχουν προσαρμοστεί στις απαιτήσεις του server και έχουν αλλάξει κατάληξη σε **.ejs** αρχεία για να μπορέσουν να ενταχθούν ως view παράθυρα. Είναι τοποθετημένα σε ένα φάκελο με όνομα views. Το EJS αρχείο σημαίνει embedded JavaScript και χρησιμοποιείται για την προβολή δυναμικών δεδομένων από τον διακομιστή. Παρόλα αυτά σε όλα τα αρχεία περιέχεται κώδικας html.

Όλα τα αρχεία ακολουθούν την ίδια φιλοσοφία. Η φιλοσοφία αυτή βασίζεται στην διάταξη της ιστοσελίδας μας σε τμήματα. Τα τμήματα αυτά έχουν φτιαχτεί σε γλώσσα μορφοποίησης CSS και καθένα από αυτά περιέχει διαφορετικές πληροφορίες για την ιστοσελίδα.

Το βασικό και κατά συνέπεια το μεγαλύτερο σε μέγεθος αρχείο, ονομάζεται **index.ejs** . Σε ένα από τα τμήματα του, στο αρχικό τμήμα, αναγράφεται σε tag header ο τίτλος της εφαρμογής. Σε όλες τις ενότητες του αρχείου έχουν χρησιμοποιηθεί διάφορα tags.

Μερικά από αυτά είναι:

** , <h1> , <b> ,
 , <button> , <form> , <h2> , <pre> , , <code> , <h3> , <p> , <input> , <label> , <u> , <hr> , <img> , <table> , <tr> , <td> , <title>**

Σε κάθε αρχείο γραμμένο σε html δεν μπορεί να λείπουν τα κύρια tags της εκ των οποίων είναι:

<html> , <head> , <body> , <meta>

Επιπλέον τα αρχεία αυτά συνδέονται με άλλα αρχεία CSS και JavaScript, τα οποία δηλώνονται στο εσωτερικό του tag head της γλώσσας με τα tag: **<link> και <script>**

Οι δύο εντολές <meta> που υπάρχουν επίσης στο head της σελίδας είναι οι

```
<meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

Και αφορούν μετά-πληροφορίες για την ιστοσελίδα που είναι ιδιαίτερα σημαντικές για την ορθή απόδοση του περιεχομένου της στα προγράμματα περιήγησης. Η πρώτη εντολή αφορά το σύνολο χαρακτήρων που μπορεί να χρησιμοποιηθεί και το UTF-8 υποστηρίζει σχεδόν όλες τις γλώσσες παγκοσμίως. Η δεύτερη εντολή αφορά την προσαρμογή της ιστοσελίδας σε κινητές συσκευές, φτιάχνοντας το πλάτος και το zoom της σελίδας ανάλογα με την συσκευή.

Στην δεξιά περιοχή της σελίδας (όταν αναφερόμαστε σε desktop και tablet συσκευές) έχουν δημιουργηθεί τα buttons με τις κατηγορίες ή αλλιώς το menu της πλατφόρμας.

```
<button class="button" , id="cMain">📌 Συνάρτηση main</button>
```

Στην κύρια περιοχή (που λέγεται main όπως θα δούμε στην ενότητα του css) της οθόνης παρέχεται όλη η πληροφορία της ιστοσελίδας, σε συνδυασμό με την μορφοποίηση που έχει

υποστεί από το αρχείο CSS. Εδώ πέρα ορίζονται εικόνες, άλλα κουμπιά, τα κουίζ της σελίδας, παραδείγματα κώδικα, ασκήσεις, θεωρία, πίνακες και γενικότερα εισάγονται όλα τα tags που υπάρχουν παραπάνω.

Σε αυτή την φωτογραφία μπορούμε να δούμε μερικά τέτοια παραδείγματα.

```
<main>
    <!-- C -->
    <div id="Cintro" , class="hide", style="position: absolute;">

        <img src="

        <br>Η <span class="magenta"><b>C</b></span> είναι μια γλώσσα, η οποία βασίζεται στην ιδέα του δομημένου προγραμματισμού κάνον
        <br><br><u><b>Δομημένος προγραμματισμός</u></b> είναι η διαίρεση του προγράμματος σε μικρότερα τμήματα, τις συναρτήσεις.
        Κάθε συνάρτηση είναι ορισμένη να επιτελεί μια συγκεκριμένη ενέργεια και μπορεί να κληθεί είτε μέσα στο κύριο πρόγραμμα,
        είτε σε κάποιο άλλο από τα υπό-προγράμματα του.
        <br><br>Η C αποτελεί μια δημοφιλή επιλογή γλώσσας για να ξεκινήσει κάποιος το ταξίδι εξερεύνησης και κατανόησης του
        προγραμματισμού και των δυνατοτήτων του. Είναι μια απλή σε επίπεδο κατανόησης γλώσσα στον κόσμο της πληροφορικής
        με ευρεία χρήση της σε εφαρμογές και λειτουργικά συστήματα. Εμείς, θα προσπαθήσουμε να κατανοήσουμε τις βασικές έννοιες
        και τεχνικές της C, καθώς και θα αναπτύξουμε απλά αποτελεσματικά προγράμματα.

        <br><br><div class="magenta"><b><u>Βασικά Χαρακτηριστικά της γλώσσας:</u></b></div>
        <br> 🔹 Όλες οι εντολές που θα προστεθούν στο πρόγραμμα μας πρέπει υποχρεωτικά να συνοδεύονται στο τέλος με το σύμβολο του ερωτ
        <br> 🔹 Ο υπολογιστής "καταλαβαίνει" μόνο την γλώσσα μηχανής, δηλαδή μόνο 0 ή 1. Όλες η εντολές που υπάρχουν στο πρόγραμμα μετα
        μεταγλωττιστή της C σε 0 ή 1.
        <br>Ο μεταγλωττιστής ή αλλιώς compiler που χρησιμοποιεί η γλώσσα C είναι ο gcc, τον οποίο τον χρησιμοποιούμε κάθε φορά που "τ
        <br> 🔹 Η C είναι case sensitive: σε κάθε λέξη τα κεφαλαία γράμματα θεωρούνται διαφορετικά από τα μικρά.
        <br> 🔹 Στην γλώσσα C υπάρχουν πολλές εντολές που μπορούμε να χρησιμοποιήσουμε για να εκτελέσουμε μια ενέργεια. Τα ονόματα τέτο
        <br><br><u><b>Δεσμευμένες λέξεις</u></b>
        είναι λέξεις που έχουν ειδικό νόημα στη γλώσσα και χρησιμοποιούνται για τη δήλωση των δομών ελέγχου του κώδικα (όπως η if, el
        άλλες συγκεκριμένες λειτουργίες, τις οποίες θα βλέπουμε σιγά σιγά κατά την διάρκεια των κεφαλαίων.
```

Τα κουμπιά που υπάρχουν κατά διαστήματα σε μερικά σημεία του κύριου τμήματος της σελίδας, είναι αυτά που μεταφέρουν τον χρήστη σε άλλες καρτέλες. Οι καρτέλες αυτές περιέχουν είτε εξωτερικές πηγές όπως βίντεο ή ιστοσελίδες, είτε ανοίγει νέο παράθυρο στον φυλλομετρητή με κάποιο από τα υπόλοιπα αρχεία γραμμένα σε html, τα οποία ακολουθούν και αυτά τα παραπάνω χαρακτηριστικά.

```
<br><button class="buttonClick" onclick="example1()">ΠΑΜΕ</button>
```

Τ υπόλοιπα αρχεία είναι τα: **example1.html, example2.html, example3.html και final.html** τα οποία είναι τα αρχεία που περιέχουν πληροφορίες σχετικά με ασκήσεις που πρέπει να επιλύσει ο χρήστης στην C. Η σχεδίαση τους ακολουθεί την ίδια πορεία.

Μέχρι στιγμής έχουμε αναφερθεί σε πέντε από τα επτά αρχεία γραμμένα σε html τα οποία ακολουθούν παρόμοια λογική και δομή ιστοσελίδας. Τα άλλα δύο αρχεία είναι τα **login.ejs** και **register.ejs** , τα οποία περιέχουν τα άλλα δύο view points της σελίδας.

Είναι και αυτά γραμμένα σε html γλώσσα και σε αντίθεση με τα υπόλοιπα αρχεία περιέχουν κομμάτια κώδικα μιας **φόρμας - form** σύνδεσης και εγγραφής στην πλατφόρμα αντίστοιχα. Οι φόρμες στέλνουν τα δεδομένα, στον διακομιστή (επεξεργασία με Node.js), σε διαδρομές που ορίζονται ως /login και /register με την μέθοδο POST κάθε φορά που ο χρήστης συνδέεται ή εγγράφεται στην φόρμα. Ταυτόχρονα εμφανίζονται και μηνύματα λάθους αν αυτά υπάρχουν κάτι που την καθιστά δυναμική ιστοσελίδα γιατί πρέπει να προσαρμόζεται σε κάθε διαφορετική ενέργεια.

Για παράδειγμα η φόρμα login ορίζεται ως:

```
<form action="/login" method="POST">
    <div class="row">

        <% if (messages.error) { %>
            <div class="alert alert-danger">
                <strong><%= messages.error %> </strong>
            </div>
        <% } %>

    <h1 class="mb-3 h3">Σύνδεση</h1>

    </div>

    <!-- Email input -->
    <div class="form-outline mb-4">
        <input type="email" name="email" id="form3Example3" class="form-control" />
        <label class="form-label" for="form3Example3">Email</label>
    </div>

    <!-- Password input -->
    <div class="form-outline mb-4">
        <input type="password" name="password" id="form3Example4" class="form-control" />
        <label class="form-label" for="form3Example4">Κωδκός Πρόσβασης</label>
    </div>


    <!-- Submit button -->
    <button type="submit" class="btn btn-primary btn-block mb-4">
    Σύνδεση
    </button>
```

Έτσι παρουσιάζεται όλη η βάση της ιστοσελίδας και ο τρόπος με τον οποίο ο χρήστης παίρνει τις πληροφορίες που θέλει να μάθει.

Περισσότερες λεπτομέρειες για την αποστολή δεδομένων και τον server υπάρχουν στο προηγούμενο κεφάλαιο, όμως θα αναφερθούμε επίσης αργότερα σε αυτό και στην ενότητα του node.js .

4.3: CSS

Η γλώσσα **CSS** όπως ξανά αναφέραμε είναι η γλώσσα μορφοποίησης της εφαρμογής. Η html χρησιμοποιεί την γλώσσα για να μπορέσει να προσαρμόσει επικεφαλίδες, κείμενα, παραγράφους, εικόνες, πίνακες, κουμπιά, ακόμη και ολόκληρη την οθόνη και πως αυτή φαίνεται σε κάθε χρήστη. Αυτός ακριβώς είναι και ο λόγος που την εντάξαμε στην «παρέα» μας.

Συνολικά, για την δημιουργία της ιστοσελίδας μας φτιάξαμε δύο ξεχωριστά αρχεία CSS. Το ένα εφαρμόζεται στο κύριο αρχείο σε html που είδαμε πριν, το index.ejs και ονομάστηκε **style.css**, ενώ το δεύτερο αρχείο μορφοποίησης εφαρμόστηκε σε όλα τα υπόλοιπα αρχεία html και ονομάστηκε **style_first.css** . Τα δύο αρχεία έχουν πολλές ομοιότητες και κοινά χαρακτηριστικά, αλλά η συνολική οπτική εικόνα του χρήστη σε αυτά τα παράθυρα είναι λίγο διαφορετική (παρουσιάζεται στο κεφάλαιο 5).

Και τα δύο αρχεία «κουμπώνουν» στο html file με το tag: **<link>**

```
<link rel="stylesheet" type="text/css" href="style.css">
```

Η πλατφόρμα έχει σχεδιαστεί να είναι responsive σε desktop, tablet και smartphones αλλάζοντας διάταξη, πλάτος, μήκος, σχεδιασμό μερικών κλάσεων (classes) που έχουν οριστεί κ.α. Όταν η συσκευή με την οποία έχει πρόσβαση ο χρήστης ξεπερνάει ένα όριο στο πλάτος (width), τότε ο τρόπος εμφάνισης της εφαρμογής αλλάζει και προσαρμόζεται στις νέες συνθήκες. Ο τρόπος με τον οποίο έγινε αυτό είναι με την εντολή **@media only screen** ως εξής:

```
@media only screen and (max-width: 480px) and (min-width: 200px){

    [class*="grid"] {
        font-size: 12px;
    }

    code {
        font-size: 10px;
    }

    [class*="container-img"]{
        grid-template-columns: 1fr;
        column-gap: 0px;
        margin-right: 1px;
    }

    img {
        width: 70%;
    }
}
```

Για παράδειγμα στην φωτογραφία αυτή βλέπουμε ότι το tag: **<code>** που χρησιμοποιούμε στην html, θέλουμε να αλλάζει το μέγεθος της γραμματοσειρά του όταν η οθόνη πρόκειται για smartphone ώστε να χωράει στο πλάτος της οθόνης και να μην εμφανίζεται κάποια μπάρα κύλισης.

```
code {
    font-family: 'Courier New', Courier, monospace;
    font-size: 14px;
}
```

Ενώ το μέγεθος έχει οριστεί ως 14px στην εντολή font-size , αλλάζει σε 10px.

Το βασικό template που ορίσαμε στην ιστοσελίδα είναι η διάταξη **grid** (display: grid;). Το grid μας επιτρέπει να χωρίζουμε την σελίδα σε διαφορετικά τμήματα. Όσον αφορά την διάταξη αυτή, για εφαρμογές desktop το grid περιλαμβάνει τα τμήματα **header, footer, aside και main**. Το header είναι η επικεφαλίδα της πλατφόρμας, το footer το υποσέλιδο της και τα τμήματα aside και main έχουν τοποθετηθεί να βρίσκονται στο κεντρικό τμήμα της σελίδας το ένα δίπλα στο άλλο, με το aside να βρίσκεται στα αριστερά της σελίδας ενώ είναι μικρότερο στο πλάτος και το main στα δεξιά της.

```
[class*="grid"] {
    display: grid;
    grid-template-columns: 1fr 4fr;
    grid-template-rows: min-content min-content auto min-content;
    grid-template-areas:
        "header header"
        "aside main"
        "footer footer";
    width: 100%;
    min-height: 100vh;
}
```

Από την άλλη όταν πρόκειται ο χρήστης να συνδέεται στην εφαρμογή από μικρότερες συσκευές, όπως smartphones, η διάταξη της εφαρμογής διατηρεί πάλι τα τμήματα με πριν που την διαχωρίζουν στο grid, όμως τα μεταφέρει το ένα κάτω από το άλλο. Το aside area από τα δεξιά που βρισκόταν δίπλα στο main πλέον βρίσκεται από πάνω του.

```css
@media only screen and (max-width: 480px) and (min-width: 200px){
    .button {
        padding: 3px 12px;
        font-size: 12px;
    }

    [class*="grid"] {
        width: 100%;
        font-size: 12px;
        grid-template-columns: 1fr;
        grid-template-rows: min-content min-content min-content min-content;
        grid-template-areas:
            "header"
            "aside"
            "main"
            "footer";
    }
```

Η ιστοσελίδα για κάθε τι που θέλει να τονίσει -όπως ένας τίτλος- χρησιμοποιεί τις επικεφαλίδες με πιο συνηθισμένα τα tag h1, h2 ή h3. Στο αρχείο css ουσιαστικά πειράζουμε την μορφή τους και άλλα τα τοποθετούμε στην μέση της οθόνης, σε άλλα κείμενα αλλάζουμε το χρώμα, σε άλλα την γραμματοσειρά. Το ίδιο μπορεί να συμβεί σχεδόν σε οποιοδήποτε tag χρησιμοποιούμε στο αρχείο html.

Για παράδειγμα:

```css
body {
    font-family: Georgia, 'Times New Roman', Times, serif;
    font-size: 16px;
}
```

```css
h2, h3 {
    font-family:'Franklin Gothic Medium', 'Arial Narrow', Arial, sans-serif;
}

h2{
    text-align: center;
}
```

Στον κώδικα μας υπάρχουν πολλές **κλάσεις** τις οποίες χρησιμοποιούμε για να περάσουμε την μορφοποίηση που θέλουμε σε πολλά στοιχεία. Αυτή είναι η δυνατότητα ομαδοποίησης που έχει η CSS.

Μια κλάση που έχει δημιουργεί είναι το **.frame** που τα στα βασικά χαρακτηριστικά του είναι το διαφορετικό χρώμα που έχει σε σχέση με την υπόλοιπη ιστοσελίδα και ένα πλαίσιο από γύρω, ώστε μέσα του να γράφονται τα παραδείγματα κώδικα που παρουσιάζονται στην θεωρία των μαθημάτων. Έχει συγκεκριμένο μέγεθος (μήκος) και σε περίπτωση που ο κώδικας δεν χωράει σε αυτό, τότε η ιδιότητα **word-wrap: break-word;** τον σπάει σε περισσότερες από μια γραμμές καθώς ο κώδικας γράφετε με την ιδιότητα **pre** που σημαίνει ότι θέλουμε να εμφανίζεται αυτούσιος στην οθόνη.

Όταν μια κλάση θέλουμε να αλλάξει κάποια ιδιότητα της σε άλλη συσκευή περικλείουμε το όνομα της σε [ ] και προσθέτουμε class*= αντί για σκέτη τελεία. Αυτό παρατηρείται στην κλάση frame αφού το μέγεθος γραμματοσειράς του κειμένου στο εσωτερικό της αλλάζει όταν πρόκειται για smartphone συσκευή.

```css
[class*="frame"] {
    border: 2px solid #000;
    padding: 20px;
    width: 80%;
    background-color: rgba(237, 246, 235, 0.884);
    padding-right: 5px;
    font-size: 12px;
    word-wrap: break-word; /* Προσθέτει αλλαγές γρα
}

.frame pre {
    white-space: pre-wrap;
    overflow-wrap: break-word;
}
```

Μια ακόμη σημαντική κλάση της εφαρμογής μας είναι τα κουμπιά του μενού (**buttons**). Κάθε κλάση ορίζεται με μία τελεία (.) και ένα όνομα συνοδευόμενο από αγκύλες ({}) , που περιέχουν τις εντολές μορφοποίησης. Για παράδειγμα:

```css
.button {
    display: inline-block;
    padding: 2px 10px;
    font-size: 16px;
    cursor: pointer;
    border: 2px solid #333;
    border-radius: 5px;
    margin: 5px 10px;
    text-align: left;
    background-color: #bd81aa;
    transition: background-color 0.3s ease;
}
```

Μία εντολή που χρησιμοποιήσαμε επίσης και αποτελεί σημαντική ιδιότητα της εφαρμογής, είναι η εντολή **visibility**. Για κάθε κουμπί που πατάει ο χρήστης από το aside area θα πρέπει να εμφανίζονται στην main περιοχή όλες οι σχετικές πληροφορίες και να αποκρύπτονται οι άσχετες. Αυτό γίνεται με την βοήθεια αυτής της εντολής και τον ορισμό της ως **hidden** αντί για visible. Ορίζονται όλα τα κείμενα να είναι εξαρχής κρυμμένα στην σελίδα και με την βοήθεια της JavaScript -που θα δούμε σε επόμενη ενότητα- εμφανίζονται και εξαφανίζονται οι πληροφορίες στην οθόνη, θέτοντας την ιδιότητα αυτή κάθε φορά visible στο κομμάτι που θέλουμε να γίνει ορατό στον χρήστη ενώ όλα τα υπόλοιπα είναι hidden.

```css
.hide {
    visibility: hidden;
}
```

Τα αρχεία css όπως είπαμε είναι δύο. Και τα δύο ακολουθούν ίδια λογική με κάποιες κοινές κλάσεις και εντολές απλώς με διαφορετική απεικόνιση των τελικών ιστοσελίδων. Το grid για παράδειγμα στο δεύτερο αρχείο δεν περιέχει καθόλου την περιοχή aside.

```css
grid-template-areas:
    "header"
    "main"
    "footer";
```

4.4: JavaScript

Το διαδραστικό μέρος της εφαρμογής υλοποιήθηκε με την γλώσσα **JavaScript**. Η χρήση της επεκτείνεται σε διάφορους τομείς, από την ανάπτυξη ιστοσελίδων μέχρι εφαρμογές για τον φυλλομετρητή και για κινητά. Αυτό την καθιστά υπεύθυνη για την ολοκλήρωση του project μας και για αυτόν τον λόγο επιλέχθηκε.

Η JavaScript είναι μια διαδικαστική γλώσσα και λειτουργεί με συναρτήσεις που εκτελούν συγκεκριμένες εργασίες. Στο δικό μας project ελέγχει μέσω συναρτήσεων απαντήσεις που έδωσε ο χρήστης στην ιστοσελίδα στα διάφορα τεστ και κουίζ που έχουν δημιουργηθεί στην html. Η html καλεί τις συναρτήσεις που υπάρχουν ανάλογα με τον σκοπό των κουίζ και η JavaScript ελέγχει τα αποτελέσματα και τα εμφανίζει στην οθόνη. Υπάρχουν δύο αρχεία js τα **script.js** και **script_first.js**. Όσο αφορά το script.js , αυτό χειρίζεται συμβάντα στο index.ejs αρχείο ενώ το δεύτερο χειρίζεται τα login.ejs και register.ejs .

Για το script.js :

Επιπλέον κάτι πολύ βασικό που συμβαίνει μέσω του αρχείου JavaScript είναι η **αποστολή δεδομένων στο server**. Τα δεδομένα στέλνονται με την μέθοδο **fetch()** χρησιμοποιώντας το πρωτόκολλο HTTP με σκοπό να σταλθούν μέσω της μεθόδου που είδαμε πολλές φορές στην εργασία, την μέθοδο **POST**.

Στην συνάρτηση fetch χρησιμοποιούμε το POST και τα δεδομένα στέλνονται σε μορφή ενός αρχείου **JSON**. Αυτό γίνεται με την δήλωση **Content-Type: application/json**. Οι απαντήσεις του χρήστη στα κουίζ είναι τα δεδομένα που υπάρχουν στην φόρμα (form) που δημιουργήσαμε στο html file, μετετρεμμένα σε μορφή JSON: **body: JSON.stringify(payload)** με το payload να είναι τα δεδομένα.

```
fetch('/submit-quiz', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json',
  },
  body: JSON.stringify(payload),
})
```

Όλη η διαδικασία της αποστολής δεδομένων γίνεται με τις συναρτήσεις **sendResultsToServer** και **sendQuizResults**. Και οι δύο συναρτήσεις κάνουν την ίδια δουλειά απλά για διαφορετικούς τύπους ασκήσεων οπότε χρειάζεται να διαφέρουν και να μην είναι μόνο μια συνάρτηση που εκτελεί αυτή την λειτουργία. Τα δεδομένα αυτά στην εφαρμογή μας, περιλαμβάνουν: id του κουίζ, το σκορ που έκανε στο κουίζ, το σύνολο ερωτήσεων και τις απαντήσεις του.

```
function sendResultsToServer(quizId, score, totalQuestions, answers) {

  const payload = {   //ti tha steiloyme
    quiz_id: quizId,
    score: score,
    totalQuestions: totalQuestions,
    answers: answers
  };
```

Στην συνέχεια ο server επεξεργάζεται τις πληροφορίες και επιστρέφει μια απάντηση που την διαβάζουμε το **.then()** , ενώ το **.catch()** διαχειρίζεται πιθανά λάθη κατά την αποστολή τους.

Ο λόγος που συμβαίνουν οι ενέργειες αποστολής και λήψης δεδομένων είναι για να αποθηκεύσουμε τις πληροφορίες στην βάση δεδομένων (στο αρχείο του node.js) και να κρατήσουμε την συνολική βαθμολογία του κάθε χρήστη σε κάθε τεστ και να του το εμφανίσουμε. Η εμφάνιση γίνεται με την συνάρτηση **displayResults** η οποία καλείτε όταν ο χρήστης πατήσει το κουμπί «Βαθμολογία» για να δει τα αποτελέσματα του, μέσω ενός **event listener**. Ο χειριστής χρησιμοποιεί και αυτός την μέθοδο **fetch()** με την **GET** αυτή την φορά να εκτελείται. Η GET κάνει μια αίτηση στον server σε ένα από τα end points όπως θα ονομάσουμε αργότερα (στο κεφάλαιο Node.js) με όνομα **/user-results** , ώστε να λάβει τα αποτελέσματα του χρήστη από τον server. Στην συνέχεια όταν ολοκληρωθούν οι ενέργειες, η εντολή **.then(response => response.json())** αναλύει τα δεδομένα της απάντησης σε μορφή πάλι JSON τα οποία εμφανίζονται στην οθόνη με την συνάρτηση **displayResults** που τα εμφανίζει από το αρχείο js σε γλώσσα html με την βοήθεια των `...` συμβόλων.

```
container.innerHTML = '';

let resultHtml = `<p>Εδώ μπορείς να δεις αναλυτικά την <b>επίδοση σου</b> κατά την διάρκεια όλων των μαθημάτων.</p><br><br>`
```

```
document.getElementById('showResultsBtn').addEventListener('click', function() {

  fetch('/user-results')

  .then(response => response.json())
  .then(data => {
      displayResults(data.results, data.totalScore);
  })

  .catch((error) => {
      console.error('Error:', error);
  });
});
```

Τα κουίζ και τεστ της εφαρμογής επίσης επεξεργάζονται από το αρχείο JavaScript με συναρτήσεις και συγκεκριμένα υπάρχουν 3 διαφορετικές συναρτήσεις που διαχειρίζονται τα διαφορετικά είδη τεστ. Υπάρχουν στο σύνολο 8 τεστ και καθένα καλεί μια από τις 3 συναρτήσεις με όρισμα της σωστές απαντήσεις του και μετά αυτές ακολουθούν την διαδικασία αποστολής στο server που αναλύσαμε πιο πάνω. Ένα παράδειγμα μιας από τις συναρτήσεις είναι το εξής:

```
function checkAnswersGeneric(quizId, correctAnswersArray, totalQuestions) {
  let correctAnswers = 0;
  let answers = [];

  correctAnswersArray.forEach((correctAnswer, index) => {
      let questionId = index + 1;
      let userAnswer = document.querySelector(`input[name="q${questionId}"]:checked`);

      if (userAnswer) {
          let isCorrect = userAnswer.value === correctAnswer;
          if (isCorrect) correctAnswers++;
          answers.push({
              question_id: questionId,
              user_answer: userAnswer.value,
              correct_answer: correctAnswer,
              is_correct: isCorrect
          });
      }
  });

  // Αποστολή των απαντήσεων στον server
  sendQuizResults(quizId, answers);
```

Με την συνάρτηση να καλείτε από το html αρχείο με ορίσματα το id του τεστ, τις σωστές απαντήσεις του και τον συνολικό αριθμό ερωτήσεων, ως:

```
<button class="button" onclick="checkAnswersGeneric(2, ['false', '2', '4'], 3)"> Υποβολή Απαντήσεων</button>
```

Άλλα κουμπιά (**buttons**) που υπάρχουν διαχειρίζονται επίσης από το js αρχείο. Όταν ο χρήστης πατήσει μία ενότητα από το μενού πρέπει να του εμφανιστεί η αντίστοιχη πληροφορία στο main area. Η εμφάνιση και απόκρυψη μηνυμάτων γίνεται με την συνάρτηση **onclick** που είδαμε και στο προηγούμενο κεφάλαιο. Κάθε φορά για να μην υπάρχει overwrite και να γράφονται συνέχεια μηνύματα το ένα πάνω στο άλλο, πρέπει μόνο ένα μήνυμα την φορά να είναι εμφανές και τα υπόλοιπα να είναι κρυφά. Για κάθε κουμπί του μενού υπάρχει και ένα αντίστοιχο μήνυμα. Επομένως η διαδικασία εμφάνισης ενός και απόκρυψης των άλλων πρέπει να συμβεί τόσες φορές όσα και τα κουμπιά ενοτήτων (υπάρχουν 14 ενότητες). Η διαχείριση ενός κουμπιού για παράδειγμα γίνεται ως:

```
document.getElementById('cVar').onclick = function() {
  document.getElementById('Cintro').style.visibility = "hidden"
  document.getElementById('CvarMSG').style.visibility = "visible"
  document.getElementById('cMainMSG').style.visibility = "hidden"
  document.getElementById('cOppMSG').style.visibility = "hidden"
  document.getElementById('cStandardMSG').style.visibility = "hidden"
  document.getElementById('cFlow1MSG').style.visibility = "hidden"
  document.getElementById('cFlow2MSG').style.visibility = "hidden"
  document.getElementById('cBreakMSG').style.visibility = "hidden"
  document.getElementById('cArraysMSG').style.visibility = "hidden"
  document.getElementById('cStringMSG').style.visibility = "hidden"
  document.getElementById('cFuncMSG').style.visibility = "hidden"
  document.getElementById('cPointersMSG').style.visibility = "hidden"
  document.getElementById('cStructMSG').style.visibility = "hidden"
  document.getElementById('cProgMSG').style.visibility = "hidden"
  document.getElementById('showResultsBtnMSG').style.visibility = "hidden"
};
```

Εδώ το 2ο μήνυμα με id **CvarMSG** ορίζεται ορατό (visible) ενώ τα υπόλοιπα ως κρυφά (hidden). Αρχικά όλα τα μηνύματα είναι κρυφά όπως τα ορίσαμε στο css αρχείο. Κάθε φορά η ορατότητα του στοιχείου που θέλουμε να είναι ορατό αλλάξει από hidden σε visible.

```
<div id="CvarMSG" , class="hide", style="position: absolute;">···
</div>
```

```
.hide {
    visibility: hidden;
}
```

Υπάρχουν όμως και **κουμπιά** που δεν εμφανίζουν κάποιο μήνυμα αλλά εκτελούν μία ενέργεια όπως η ανακατεύθυνση του χρήστη σε κάποιο εξωτερικό σάιτ ή η μεταφορά του σε κάποια από τις ασκήσεις που φτιάξαμε σε άλλο αρχείο html. Αυτό γίνεται πολύ εύκολα με την ιδιότητα **window.open** .

```
function final() {
  window.open('final.html', '_blank');
}

function Video() {
  window.open('https://www.youtube.com/watch?v=KYxLEDF6kjs', '_blank');
}
```

Για το script_first.js :

Ο κώδικας του αρχείου αυτού εφαρμόζεται στο **login.ejs** για την εμφάνιση στοιχείων μόλις το ποντίκι ακουμπήσει το κουμπί (hover), χωρίς να το πατήσει. Χρησιμοποιεί το **DOM** για να γίνει η διαδραστική λειτουργία.

```
document.addEventListener('DOMContentLoaded', () => {
    const buttons = document.querySelectorAll('.button');
```

Υπάρχουν δύο **event listeners** για τα γεγονότα **mouseover** και **mouseout** όταν ο χρήστης δηλαδή πάει πάνω στο κουμπί με το ποντίκι του και όταν φεύγει εκτός αυτού. Το στοιχείο που εμφανίζεται είναι ένα **menu**. Όταν ο χρήστης περνάει το ποντίκι πάνω στο μενού αυτό παραμένει εμφανές και δεν εξαφανίζεται κατευθείαν.

```
buttons.forEach(button => {
    button.addEventListener('mouseover', () => {
        const menu = button.nextElementSibling;
        if (menu) {
            menu.style.display = 'block';
        }
    });

    button.addEventListener('mouseout', () => {
        const menu = button.nextElementSibling;
        if (menu) {
            menu.style.display = 'none';
        }
    });

    const menu = button.nextElementSibling;
    if (menu) {
        menu.addEventListener('mouseover', () => {
            menu.style.display = 'block';
        });

        menu.addEventListener('mouseout', () => {
            menu.style.display = 'none';
        });
    }
});
```

Ο κώδικας JavaScript που ασχολείται κυρίως με frontend είναι αυτός. Ωστόσο στην ίδια γλώσσα προγραμματισμού ανήκει το εργαλείο Node.js το οποίο διαχειρίζεται την λειτουργία του server. Προτιμήσαμε να το τοποθετήσουμε σε διαφορετική ενότητα. Παρακάτω θα εξηγήσουμε πως λειτουργεί.

4.5: NODE.js

Για την δημιουργία του project χρειάστηκε να υπάρχει ένα εργαλείο διαχείρισης της ιστοσελίδας το οποίο θα επικοινωνεί με τον **server** και θα στέλνει ή θα λαμβάνει δεδομένα και πληροφορίες από αυτόν και την βάση δεδομένων. Για τον σκοπό αυτόν χρησιμοποιήσαμε το εργαλείο Node.js .

Το **Node.js** πρόκειται για ένα εργαλείο φτιαγμένο σε γλώσσα JavaScript, λειτουργώντας από την μεριά του server και επιτρέποντας την εκτέλεση κώδικα στον διακομιστή αντί για τον φυλλομετρητή. Περισσότερες πληροφορίες για την γενική λειτουργία του μπορούν να βρεθούν στο κεφάλαιο 3.

Η **θύρα – port** είναι το αναγνωριστικό εφαρμογών που εκτελούνται σε έναν υπολογιστή ή έναν server. Όταν αυτός επικοινωνεί μέσω του διαδικτύου, οι θύρες βοηθούν στον καθορισμό της εφαρμογής που πρέπει να σταλθούν τα δεδομένα. Ο καθορισμός της θύρας είναι απαραίτητος για μια εφαρμογή, διότι ο server μπορεί να εξυπηρετεί πολλές διαφορετικές υπηρεσίες. Στον κώδικα μας η θύρα «ακούει» την **4000**. Αυτό σημαίνει ότι ο κάθε χρήστης που θέλει να αποκτήσει πρόσβαση στην εφαρμογή, θα πρέπει να συνδεθεί στην θύρα 4000 του server με μία **τοπική διεύθυνση IP (local host)** .

```
const port = 4000;
```

```
app.listen(port, function(){
  console.log("server listening at " + port)
})
```

Το αρχείο διαχείρισης του server με το εργαλείο Node.js έχει ονομαστεί **server.js** και το έχουμε ορίσει ως προεπιλεγμένο αρχείο στο **package.json** του έτοιμου αρχείο node.

```
"scripts": {
  "devStart": "nodemon server.js"
```

Επομένως, ο server μας ξεκινάει να ακούει στην θύρα 4000 ενεργοποιώντας τον από το command prompt κατευθείαν με την εντολή: **npm run devStart**

```
C:\Users\tgeor\Desktop\code\WWW>npm run devStart

> userauth@1.0.0 devStart
> nodemon server.js

[nodemon] 2.0.22
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node server.js`
server listening at 4000
```

Επιπλέον, στον κώδικα φορτώνεται το αρχείο **.env** για να χρησιμοποιηθούν οι μεταβλητές περιβάλλοντος για ευκολία στην διαχείριση διαμορφώσεων που μπορεί να προκύψουν στο αρχείο.

Το .env περιλαμβάνει τα δεδομένα διαχείρισης της βάσεις δεδομένων που θα δούμε σε λίγο και προστίθεται στο server.js με την εντολή:

```
JS server.js > ...
1   if (process.env.NODE_ENV !== "production") {
2     require("dotenv").config()
3   }
```

Όλα τα αιτήματα σύνδεσης, αποστολής, αποθήκευσης, λήψεις δεδομένων από και προς την βάση δεδομένων, η αλλαγή των views που είδαμε στην ενότητα του HTML, η υποστήριξη στατικών αρχείων css και JavaScript, η αρχικοποίηση του Passport για την ταυτοποίηση του χρήστη κ.α. πραγματοποιούνται με την χρήση του **Express.js** , ενός δημοφιλούς framework για την δημιουργία web εφαρμογών βάση του εργαλείου Node.js .

4. 5.α: EXPRESS.js

Αρχικοποίηση:

Όπως αναφέραμε το εργαλείο express χρησιμοποιείται στην δημιουργία εφαρμογών και κάνει την χρήση του server πιο εύκολη. Παρέχει ένα σύνολο εργαλείων και λειτουργιών για την διαχείριση διακομιστών, κάνοντας την πιο απλή και κατανοητή για τον προγραμματιστή.

```
const express = require("express")
const app = express()
const port = 4000;
```

Ο server είπαμε πολλές φορές ότι λειτουργεί με την επεξεργασία αιτημάτων και συγκεκριμένα με **HTTP** αιτήματα. Το σύστημα δρομολόγησης (routing) του express είναι πολύ ισχυρό, επιτρέποντας την κατηγοριοποίηση των αιτημάτων με βάση το **URL** και την μέθοδο που χρησιμοποιεί ο server (**GET, POST**).

```
app.use(express.urlencoded({extended: false}))
app.use(methodOverride("_method"))

app.use(express.json());
```

Στατικά αρχεία:

Χάρη στην δυνατότητα των συναρτήσεων που μπορούν να εκτελούνται μεταξύ του κάθε αιτήματος του χρήστη και των απαντήσεων του, ο server χρησιμοποιεί **middleware**. Ένα τέτοιο παράδειγμα είναι η διαχείριση αρχείων.

Η υποστήριξη στατικών αρχείο CSS και JavaScript γίνεται με την εντολή **express.static**

```
app.use(express.static('public'));
```

Passport και ταυτοποίηση χρήστη:

Επιπρόσθετα, το αρχείο μας έχει άμεση σχέση με την βάση δεδομένων. Για την σύνδεση χρησιμοποιείται το **pg module**, όπου η πληροφορίες της βάσης λαμβάνονται από το .env αρχείο.

```
const pool = new Pool({
  user: process.env.DB_USER,
  host: process.env.DB_HOST,
  database: process.env.DB_NAME,
  password: process.env.DB_PASSWORD,
  port: process.env.DB_PORT
});
```

Οι πληροφορίες αυτές είναι βασικές για τον server μας για την διαδικασία της ταυτοποίησης του χρήστη κατά την διαδικασία της σύνδεσης του στην πλατφόρμα. Στο εργαλείο του Node.js υπάρχει μια βιβλιοθήκη με όνομα **passport** που επιτελεί την διαδικασία ταυτοποίησης των χρηστών. Υπάρχει το αρχείο **passport-config.js** το οποίο περιέχει την διαμόρφωση της

```
const passport = require("passport")
const initializePassport = require("./passport-config")
const flash = require("express-flash")
const session = require("express-session")
const methodOverride = require("method-override")
```

ταυτοποίησης, με το εργαλείο **express-session** για την διατήρηση της αποθήκευσης των χρηστών στην εφαρμογή, ακόμη και όταν αυτός αλλάζει σελίδα. Το **express-flash** είναι ένα ακόμη middleware για την εμφάνιση προσωρινών μηνυμάτων στον χρήστη για παράδειγμα όταν γίνει ανεπιτυχής σύνδεση του στην πλατφόρμα.

```
app.use(flash())
app.use(session({
  secret: process.env.SESSION_SECRET,
  resave: false,
  saveUninitialized: false
}))
app.use(passport.initialize())
app.use(passport.session())
app.use(methodOverride("_method"))
```

Έπειτα στον κώδικα έχουν οριστεί οι συναρτήσεις:

Το **initializePassport()** καλείται για να ρυθμίσει το local strategy που χρησιμοποιείται κατά το login των χρηστών. Η μία στρατηγική είναι η αναζήτηση του χρήστη ,μέσω **email**. Όταν πάει ο χρήστης να κάνει **login** με το email του δημιουργείται ένα **query** στην βάση δεδομένων για να βρεθεί ο χρήστης που έχει αυτό το email. Αν βρεθεί ο χρήστης, τον επιστρέφει. Αντίθετα, η δεύτερη στρατηγική έχε να κάνει με την αναγνώριση του χρήστη αν είναι συνδεδεμένος μέσω του id του.

```
initializePassport(
  passport,
  async email => {
    const res = await pool.query("SELECT * FROM users WHERE email = $1", [email]);
    const user = res.rows[0];
    return user;
  },
  async id => {
    const res = await pool.query("SELECT * FROM users WHERE id = $1", [id]);
    return res.rows[0];
  }
);
```

Η στρατηγική αναζήτησης του χρήστη, τον πιστοποιεί βάσει τα στοιχεία που θα υποβάλει (email και κωδικό πρόσβασης). Η πιστοποίηση του κωδικού πρόσβασης ορίζεται στο αρχείο passport-config.js ως

```
const LocalStrategy = require("passport-local").Strategy;
```

Στην συνάρτηση **initialize()** γίνεται πρώτα αναζήτηση του χρήστη στην βάση δεδομένων με το email και αν ο χρήστης δεν βρεθεί επιστρέφει μήνυμα λάθους. Στην συνέχεια αν βρεθεί το email προχωρά στην ταυτοποίηση κωδικού πρόσβασης με την βοήθεια της βιβλιοθήκης **bcrypt** (για να είναι κρυπτογραφημένος ο κωδικός). Αν οι κωδικοί είναι ίδιοι καλείται η συνάρτηση done με τον χρήστη και σε αντίθετη περίπτωση επιστρέφεται μήνυμα λάθους.

```
function initialize(passport, getUserByEmail) {
  const authenticateUsers = async (email, password, done) => {

    try {
      //αναζητηση του χρηστη με email
      const user = await getUserByEmail(email);
      if (user == null) {
        return done(null, false, { message: "No user found with that email" });
      }

      //σύγκριση κωδικου που εδωσε με τον κωδικο στη βαση δεδομενων
      const isMatch = await bcrypt.compare(password, user.password);

      if (isMatch) {
        return done(null, user);
      } else {
        return done(null, false, { message: "Password Incorrect" });
      }
    }
    catch (e) {
      console.log(e);
      return done(e);
    }
  };
```

Όταν πιστοποιείται επιτυχώς, το passport αποθηκεύει το id του στο session για να μπορεί να τον αναγνωρίσει αργότερα.

```
passport.use(new LocalStrategy({ usernameField: 'email' }, authenticateUsers));
passport.serializeUser((user, done) => done(null, user.id));
```

Ενώ το **deserializeUser** καλείται όταν χρειάζεται να ανακτήσει πληροφορίες για τον χρήστη που είναι ήδη συνδεδεμένος βάση του id. Αυτό γίνεται πάλι με ένα query σε **SQL** με την εντολή **SELECT * FROM users WHERE id = …** και επιστρέφει τον χρήστη στο passport με την συνάρτηση **done** που είπαμε και νωρίτερα.

Οι πληροφορίες αυτές συνδέονται με το αρχείο server.js του node και με ένα middleware ελέγχει αν ο χρήστης είναι ή όχι συνδεδεμένος (**checkAuthenticated και checkNotAuthenticated**). Είναι μέθοδοι που παρέχονται από το passport και αν ο χρήστης είναι συνδεδεμένος η εντολή **next()** ανακατευθύνει την εφαρμογή στην αρχική σελίδα, ενώ αν δεν είναι συνδεδεμένος ανακατευθύνει την εφαρμογή στην σελίδα του login.

```
function checkAuthenticated(req, res, next){
  if(req.isAuthenticated()){
      return next()
  }
  res.redirect("/login")
}

function checkNotAuthenticated(req, res, next){
  if(req.isAuthenticated()){
      return res.redirect("/")
  }
  next()
}
```

Η διαδικασία του login καλεί την local strategy αναζήτησης του χρήστη που αναλύσαμε και ελέγχει καλώντας τις δύο προηγούμενες συναρτήσεις (checkAuthenticated και checkNotAuthenticated) εάν έγινε επιτυχής πιστοποίηση του χρήστη και τον κατευθύνει στο αντίστοιχο view point.

```
app.post("/login", checkNotAuthenticated, passport.authenticate("local", {
  successRedirect: "/",
  failureRedirect: "/login",
  failureFlash: true
}))
```

Τέλος όσον αφορά τις πιστοποιήσεις υπάρχει και το register route για την διαδικασία εγγραφής ενός νέου χρήστη στην πλατφόρμα με την εντολή **POST**. Χρησιμοποιεί το **bcrypt.hash** για την κρυπτογράφηση του κωδικού πρόσβαση ώστε να μην αποθηκεύεται κατευθείαν στην βάση δεδομένων, προσφέροντας ασφάλεια στην εφαρμογή. Με ένα query πάλι σε SQL προσθέτει τον νέο χρήστη στην βάση με πεδία name, email και password. Αν όλα πάνε καλά τότε ο χρήστης μεταφέρετε στην σελίδα του login για να συνδεθεί. Αλλιώς αν υπάρξει κάποιο πρόβλημα εμφανίζονται μηνύματα λάθους.

```
app.post("/register", checkNotAuthenticated, async (req, res) => {

  try {
      const hashedPassword = await bcrypt.hash(req.body.password, 10)

      await pool.query(
        "INSERT INTO users (name, email, password) VALUES ($1, $2, $3)",
        [req.body.name, req.body.email, hashedPassword]
      );

      res.redirect("/login")

  } catch (e) {
      console.log(e);
      res.redirect("/register")
  }
})
```

View points της εφαρμογής:

Τα view point είναι η βασική πλοήγηση μεταξύ των σελίδων **login, register και της κύριας σελίδας της εφαρμογής.**

Η διαδρομή route **(/)** εξυπηρετεί την κύρια σελίδα που βρίσκεται στο αρχείο **index.ejs** και μεταβαίνει με την εντολή **GET**.

```
app.get('/', checkAuthenticated, (req, res) => {
  res.render("index.ejs", {name: req.user.name})
})
```

Η σελίδα σύνδεσης: **/login** υπάρχει στο **login.ejs**

```
app.get('/login', checkNotAuthenticated, (req, res) => {
  res.render("login.ejs")
})
```

Η σελίδα εγγραφής: **/register** υπάρχει στο αρχείο **register.ejs**

```
app.get('/register', checkNotAuthenticated, (req, res) => {
  res.render("register.ejs")
})
```

Επιπλέον υπάρχει και μια διαδρομή που ακολουθεί η εφαρμογή μόλις γίνει το **logout** του χρήστη, η οποία σε ανακατευθύνει ξανά στο login form.

```
app.delete("/logout", (req, res) => {
  req.logout(req.user, err => {
      if (err) return next(err)
      res.redirect("/")
  })
})
```

End points:

Στον κώδικα μας υπάρχουν δύο end points τα οποία χρησιμεύουν στην επικοινωνία του server με την βάση δεδομένων. Το πρώτο είναι υπεύθυνο για την αποθήκευση δεδομένων (τις πληροφορίες των κουίζ) και το δεύτερο για την λήψη δεδομένων του χρήστη (την επίδοση του στα κουίζ).

Αφού η μία διαδρομή **(/submit-quiz)** χρησιμοποιείται στην υποβολή των πληροφοριών, γίνεται με την μέθοδο **post**. Παίρνει το id του συνδεδεμένου χρήστη από το session και δημιουργεί μια νέα εγγραφή στον πίνακα με τις απαντήσεις του στα τεστ. Για κάθε μία από τις απαντήσεις ελέγχει αν υπάρχει ήδη κάποια απάντηση για το συγκεκριμένο τεστ και για την συγκεκριμένη ερώτηση. Αν δεν βρεθεί απάντηση τότε αποθηκεύεται η νέα στην βάση δεδομένων και προσθέτει μια νέα εγγραφή στον πίνακα **quiz_answers** που υπάρχει στην βάση όπως θα δούμε σε λίγο. Εφόσον το πρωτόκολλο για το οποίο μιλάμε είναι HTTP, το αποτέλεσμα αν όλα βαίνουν επιτυχώς θα επιστρέψει μια τιμή 200 (OK) και σε αντίθετη περίπτωση την τιμή 500 (server error). Τα στοιχεία για τα κουίζ στέλνονται από το frontend στο node.js με το εργαλείο **AJAX** και το **Fetch API** όπως αναλύθηκε στο υπό-κεφάλαιο της JavaScript.

```
app.post('/submit-quiz', checkAuthenticated, async (req, res) => {
  const userId = req.user.id;
  const { quiz_id, answers } = req.body;

  try {
      for (let answer of answers) {
          // Έλεγχος αν υπάρχει ήδη απάντηση για το ίδιο quiz_id, question_id και user_id
          const result = await pool.query(
              "SELECT * FROM quiz_answers WHERE user_id = $1 AND quiz_id = $2 AND question_id = $3",
              [userId, quiz_id, answer.question_id]
          );

          if (result.rows.length === 0) {
              await pool.query(
                  "INSERT INTO quiz_answers (user_id, quiz_id, question_id, user_answer, correct_answer, is_correct) VALUES ($1, $2, $3, $4, $5, $6)",
                  [userId, quiz_id, answer.question_id, answer.user_answer, answer.correct_answer, answer.is_correct]
              );
          }
      }

      res.status(200).json({ message: "Results saved successfully." });
  } catch (err) {
      console.error(err);
      res.status(500).json({ error: "An error occurred." });
  }
});
```

Η δεύτερη διαδρομή **(/user-results)** χρησιμοποιείται για την λήψη δεδομένων και γίνεται με την μέθοδο **GET**. Παίρνει το id του συνδεδεμένου χρήστη από το session και με ένα **SQL query** λαμβάνει τις απαντήσεις του χρήστη από τον πίνακα **quiz_answers** της βάσης δεδομένων. Από τον πίνακα επιστρέφονται όλες οι στήλες φιλτράροντας τα αποτελέσματα βάση του user id. Με ένα δεύτερο SQL query υπολογίζει τον αριθμό των σωστών απαντήσεων του χρήστη χρησιμοποιώντας το **COUNT(*)** για να τις μετρήσει. Στον πίνακα υπάρχει μια στήλη με όνομα **is_correct** τύπου Boolean και όπου είναι true ο χρήστης έχει απαντήσει σωστά. Τα αποτελέσματα επιστρέφονται σε μορφή ενός **json file** με μια λίστα των απαντήσεων και τον συνολικό τους αριθμό. Με την μέθοδο **Fetch** πάλι στέλνονται τα αποτελέσματα στο frontend και εμφανίζονται στον χρήστη. Η διαδικασία αυτή υλοποιείται σε μια συνάρτηση το αρχείο script.js και την εξηγήσαμε προηγουμένως.

```
app.get('/user-results', checkAuthenticated, async (req, res) => {
  const userId = req.user.id;

  try {
      const results = await pool.query(
          "SELECT quiz_id, question_id, user_answer, correct_answer, is_correct FROM quiz_answers WHERE user_id = $1",
          [userId]
      );

      const totalScore = await pool.query(
          "SELECT COUNT(*) AS correct_answers FROM quiz_answers WHERE user_id = $1 AND is_correct = true",
          [userId]
      );

      res.json({
          results: results.rows,
          totalScore: totalScore.rows[0].correct_answers
      });
  } catch (err) {
      console.error(err);
      res.status(500).json({ error: "An error occurred." });
  }
});

app.listen(port, function(){
  console.log("server listening at " + port)
})
```

Παρακάτω θα πούμε μερικές πληροφορίες για την βάση δεδομένων που δημιουργήσαμε.

4.6: PostgreSQL

Το **PostgreSQL** είναι ένα ανοιχτού κώδικα σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων που χρησιμοποιείται ευρέως για την μόνιμη αποθήκευση πληροφοριών. Για να συνδεθεί με το εργαλείο του **Node** χρησιμοποιείται η βιβλιοθήκη **pg** που παρέχει ένα API για την αλληλεπίδραση αυτών των δύο. Η σύνδεση με την βάση στο αρχείο του node.js είδαμε και σε προηγούμενη ενότητα ότι γίνεται ως:

```
const pool = new Pool({
  user: process.env.DB_USER,
  host: process.env.DB_HOST,
  database: process.env.DB_NAME,
  password: process.env.DB_PASSWORD,
  port: process.env.DB_PORT
});
```

Όπου οι πληροφορίες για το **όνομα, τον host, τον χρήστη, τον κωδικό και την θύρα** παρέχονται από το .**env** αρχείο στο οποίο τα έχουμε ορίσει ως:

```
SESSION_SECRET=secret

DB_USER=postgres
DB_HOST=localhost
DB_NAME=myapp
DB_PASSWORD=g20062002g
DB_PORT=5432
```

Για την χρήση του PostgreSQL χρησιμοποιήσαμε την εφαρμογή **pgAdmin4** και γράψαμε τις εντολές σε **SQL** στο **cmd** με την εντολή:

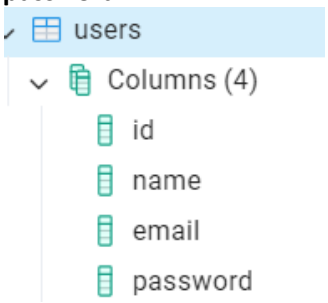
```
psql -U postgres -d myapp -h localhost -p 5432
```

Όπου ο server του PostgreSQL πρέπει να είναι ανοιχτός και συνδεδεμένος στην προεπιλεγμένη από τις ρυθμίσεις του, **θύρα 5432**.

4. 6.α: pgAdmin4

Στο σύνολο υπάρχουν **δύο πίνακες στην βάση δεδομένων** και τους διαχειριζόμαστε με την εφαρμογή pgAdmin4.

Ο **πρώτος πίνακας** λέγεται **user** και περιέχει τέσσερις στήλες δεδομένων τις: **id, email, name και password.**



Η στήλη **id** περιλαμβάνει τον αύξων αριθμό κάθε χρήστη που αυξάνεται κατά ένα κάθε φορά που εγγράφεται ένας νέος χρήστης στην πλατφόρμα. Είναι το πρωτεύον κλειδί του πίνακα και αυτό σημαίνει ότι από αυτό το πεδίο αναγνωρίζεται ο συγκεκριμένος χρήστης.

Η στήλη **name** περιλαμβάνει το ονοματεπώνυμο του χρήστη που εκείνος δίνει στην φόρμα κατά την εγγραφή του.

Η στήλη **email** περιλαμβάνει το email του κάθε χρήστη αντίστοιχα και είναι μια από τις δύο στήλες τις οποίες χρησιμοποιούμε για την ταυτοποίηση του στην εφαρμογή.

Η στήλη **password** τέλος, περιλαμβάνει το κρυπτογραφημένο κωδικό πρόσβασης του χρήστη που έχει ορίσει αυτός κατά την διαδικασία της εγγραφής του και αποτελεί το δεύτερο στοιχείο για την ταυτοποίηση του κατά την είσοδο του.

Για την εμφάνιση του πίνακα δημιουργούμε εκείνη την ώρα ένα **query** μέσω του query tool της εφαρμογής και σε **SQL** γράφουμε την εντολή:

```
SELECT * FROM users;
```

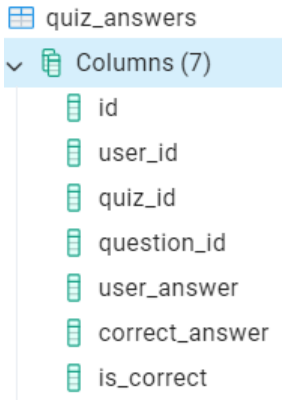
Και μας εμφανίζονται όλοι οι χρήστες του πίνακα.

	id [PK] integer	name character varying (255)	email character varying (255)	password character varying (255)
1	1	Georgia Toumpa	gwgw.toumpa@gmail.com	$2b$10$x4qH.IqDfT4WE7ZB7QB1Te8fAJNOfHWsITqRTl4n1l41sxoCZzZaa
2	2	ΜΑΡΙΑ ΑΡΣΕΝΙΔΟΥ	axil@gmail.com	$2b$10$nFxN44SoHS7qSVho7IzRQ.E20BkEYYDIOuxTs.9b4COURbbnzl4R.
3	3	axileas toubas	tgeorgia.gogo@gmail.com	$2b$10$EEOdtEbW1S9V37Bn4fZhaeUqz28c5eEpf/wxdyszNbs19sFRvcoXe

Οι εντολές σε SQL για τον πίνακα:

```
CREATE TABLE IF NOT EXISTS public.users
(
    id integer NOT NULL DEFAULT nextval('users_id_seq'::regclass),
    name character varying(255) COLLATE pg_catalog."default" NOT NULL,
    email character varying(255) COLLATE pg_catalog."default" NOT NULL,
    password character varying(255) COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT users_pkey PRIMARY KEY (id),
    CONSTRAINT users_email_key UNIQUE (email)
)
```

Ο **δεύτερος πίνακας** είναι αυτός που κρατάει τα δεδομένα του χρήστη για την επίδοση του στα κουίζ της εφαρμογής. Ονομάζεται **quiz_answers**. Συνδέεται με μια σχέση με τον πίνακα users περνώντας το user id από τον έναν πίνακα στον άλλον. Ο πίνακας περιέχει επτά διαφορετικές στήλες: **id, user_id, quiz_id, question_id, user_answer, correct_answer, is_correct.**



Η στήλη **id** περιέχει τον Α/Α του πίνακα συμπεριλαμβανομένων όλων των χρηστών της πλατφόρμας. Είναι το πρωτεύον κλειδί του πίνακα.

Η στήλη **user id** είναι η ίδια που υπάρχει και στον πίνακα users με όνομα id και είναι το μοναδικό αναγνωριστικό του κάθε χρήστη. Αποτελεί το δευτερεύον κλειδί του πίνακα.

Η στήλη **quiz id** περιλαμβάνει ένα επίσης μοναδικό αναγνωριστικό σε αύξων σειρά για κάθε κουίζ που φτιάξαμε.

Η στήλη **question id** αντίστοιχα είναι ο αριθμός κάθε ερώτησης σε συγκεκριμένο κουίζ κάθε φορά, στο κουίζ που υπάρχει το quiz id.

Η στήλη **user answer** είναι η απάντηση που έδωσε ο χρήστης στην φόρμα για το συγκεκριμένο question id που υπάρχει στο quiz id.

Η στήλη **correct answer** είναι η σωστή απάντηση της συγκεκριμένης ερώτησης του question id για το quiz id.

Η στήλη **is correct** είναι μια μεταβλητή τύπου Boolean η οποία συγκρίνει μέσω του κώδικα για κάθε ερώτηση των κουίζ, εάν η απάντηση που έδωσε ο χρήστης (στο user_answer) είναι ίδια με την σωστή απάντηση της ερώτησης (στο correct_answer). Αν ταυτίζονται τότε το πεδίο παίρνει την τιμή true και σε αντίθετη περίπτωση την τιμή false.

Για την εμφάνιση του πίνακα δημιουργούμε εκείνη την ώρα ένα **query** μέσω του query tool της εφαρμογής και σε **SQL** γράφουμε την εντολή:

```
SELECT * FROM quiz_answers;
```

Ο πίνακας με τα στοιχεία μοιάζει κάπως έτσι:

id [PK] integer	user_id integer	quiz_id integer	question_id integer	user_answer character varying (255)	correct_answer character varying (255)	is_correct boolean
56	3	6	4	1	2	false
57	3	6	5	4	1	false
58	3	6	6	4	1	false
59	1	1	1	true	true	true
60	1	1	2	false	false	true
61	1	1	3	3	3	true
62	1	2	1	true	false	false
63	1	2	2	2	2	true

Οι εντολές σε SQL για τον πίνακα:

```
CREATE TABLE IF NOT EXISTS public.quiz_answers
(
    id integer NOT NULL DEFAULT nextval('quiz_answers_id_seq'::regclass),
    user_id integer NOT NULL,
    quiz_id integer NOT NULL,
    question_id integer NOT NULL,
    user_answer character varying(255) COLLATE pg_catalog."default",
    correct_answer character varying(255) COLLATE pg_catalog."default",
    is_correct boolean,
    CONSTRAINT quiz_answers_pkey PRIMARY KEY (id),
    CONSTRAINT fk_user FOREIGN KEY (user_id)
        REFERENCES public.users (id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)
```

Αυτά είναι όλα τα εργαλεία που χρησιμοποιούνται στον κώδικα για την διαχείριση του server στο back end, τα στατικά αρχεία στο front end, η βάση δεδομένων και τα views.

Όλα αυτά συνδυάζονται μεταξύ τους για να αναπτυχθεί το τελικό αποτέλεσμα της διαδικτυακής εφαρμογής εκμάθησης γλωσσών προγραμματισμού που δημιουργήσαμε για την εργασία.

Η παρουσίαση του τελικού αποτελέσματος που είναι ορατό στον φυλλομετρητή αναλύεται στο επόμενο κεφάλαιο.

ΚΕΦΑΛΑΙΟ 5 ΕΠΙΔΕΙΞΗ ΕΦΑΡΜΟΓΗΣ

5.1: ΛΕΙΤΟΥΡΓΙΑ ΕΦΑΡΜΟΓΗΣ

Για τον σκοπό της εργασίας αυτής αναπτύχθηκε μια εφαρμογή με στόχο να παρέχει πληροφορίες, να διδάσκει, να ελέγχει και να παραθέτει αποτελέσματα για πληροφορίες, όσον αφορά την εκμάθηση γλωσσών προγραμματισμού. **Επιλέξαμε την γλώσσα C ως το κεντρικό θέμα της πλατφόρμας** και προσπαθήσαμε να καλύψουμε όσο το δυνατόν περισσότερα θέματα πάνω σε αυτήν. Η εφαρμογή αρχικά δίνει κάποιες βασικές πληροφορίες πιο γενικά για την γλώσσα C όπως ότι όλες οι εντολές τις γλώσσας τελειώνουν με το σύμβολο του ελληνικού ερωτηματικού (;), πως μπορεί κάποιος να κατεβάσει ένα περιβάλλον για να γράψει κώδικα στην γλώσσα αυτήν και διάφορα άλλα.

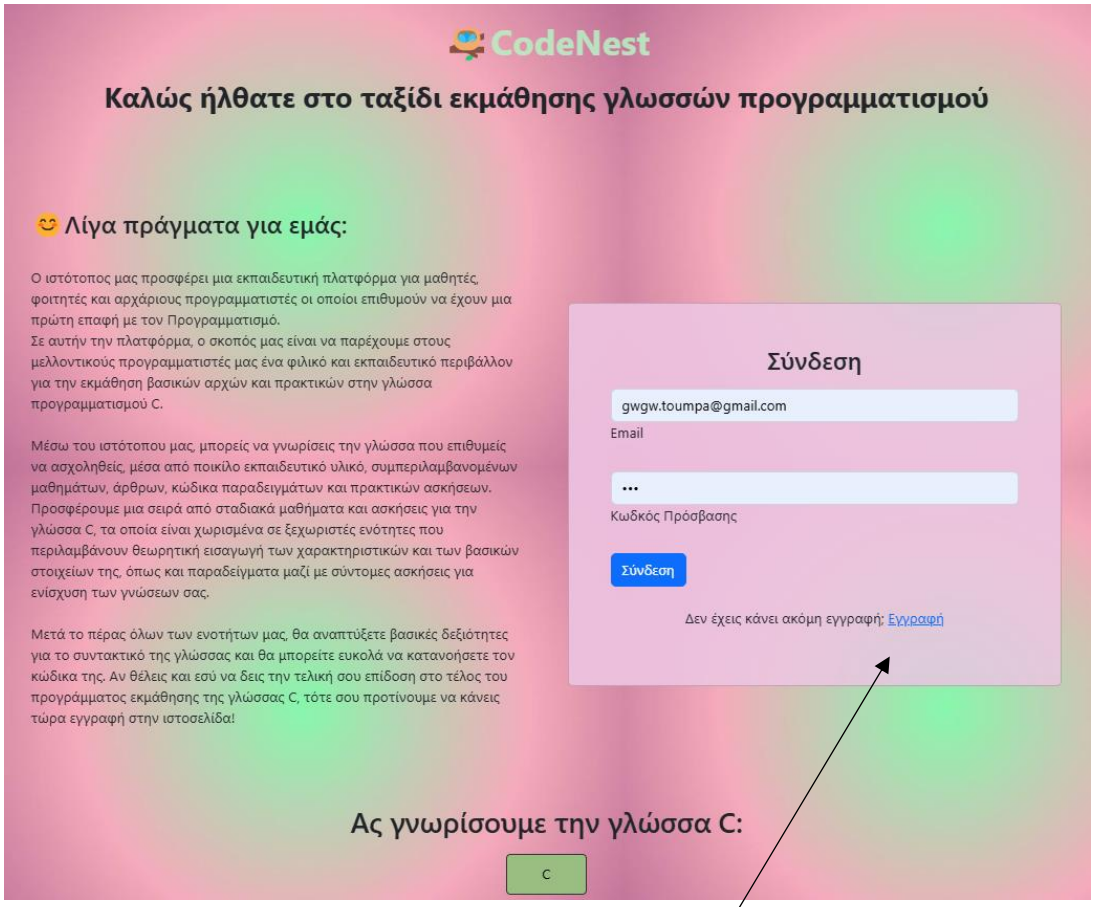
Η εφαρμογή μας **απευθύνεται κυρίως σε νέους και νέες** που θέλουν να αρχίσουν το ταξίδι τους στην εκμάθηση της C και γενικότερα σε **νέους προγραμματιστές** και άτομα που θέλουν να ασχοληθούν με τον προγραμματισμό.

Τα κεφάλαια περιλαμβάνουν λεπτομερή παρουσίαση του θεωρητικού μέρους της γλώσσας και συγκεκριμένα, είναι χωρισμένη σε δώδεκα διαφορετικές ενότητες. Λόγω όμως, ότι το κοινό στο οποίο απευθύνεται η ιστοσελίδα, είναι αρχάριοι προγραμματιστές που δεν έχουν κάποια εμπειρία πάνω στην C, αποφασίστηκε η ύλη της να μην εμβαθύνει ιδιαίτερα σε πολύ εξειδικευμένες και περίπλοκες έννοιες, αλλά να φτάσει μέχρι ένα ικανοποιητικό σημείο της.

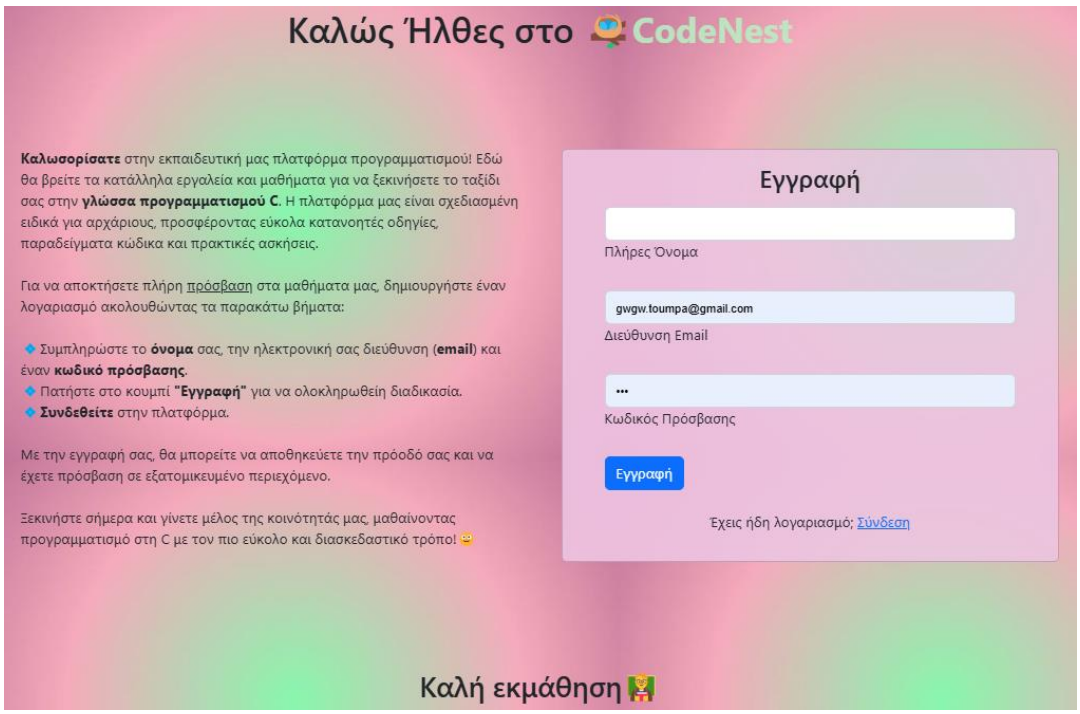
Επομένως, ενότητες της ύλης αποτελούν οι εξής:

- Συνάρτηση main
- Μεταβλητές και Σταθερές
- Τελεστές και Εκφράσεις
- Βασικές Εντολές και Βιβλιοθήκη stdio.h
- Δομές Ελέγχου Ροής (1)
- Δομές Ελέγχου Ροής (2)
- Break και Continue
- Πίνακες
- Αλφαριθμητικά Strings
- Συναρτήσεις
- Δείκτες
- Δομές Struct

Για να μπορέσει ένας χρήστης να περιηγηθεί στις ενότητες θα πρέπει πρώτα να συνδεθεί στην εφαρμογή. **Η σελίδα σύνδεσης είναι το πρώτο παράθυρο** που συναντάει μπροστά του, όπου παρέχονται μερικές πληροφορίες σχετικές με την ιστοσελίδα και υπάρχει επίσης μια φόρμα για να συμπληρώσει τα στοιχεία του: **email** και **κωδικός πρόσβασης**. Αφού ολοκληρώσει την συμπλήρωση των στοιχείων του και πατήσει το κουμπί **«Σύνδεση»** , θα μεταφερθεί στην κύρια σελίδα της εφαρμογής μας.



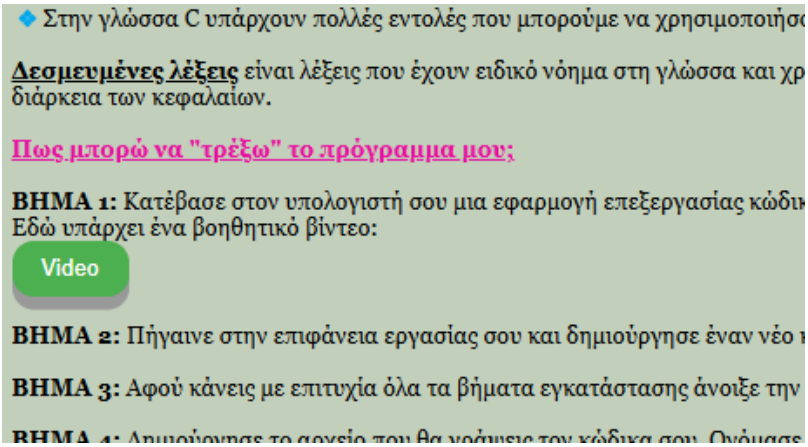
Στην περίπτωση που ο χρήστης συνδέεται για πρώτη φορά στην ιστοσελίδα, θα χρειαστεί να εγγραφεί σε αυτήν πατώντας πάνω στο URL της λέξης **«Εγγραφή»** για να μεταβεί στην **σελίδα εγγραφής** του. Σε αυτήν την σελίδα υπάρχει πάλι μια φόρμα συμπλήρωσης των στοιχείων: **Ονοματεπώνυμο** , **email** και **κωδικός πρόσβασης**. Αφού λοιπόν συμπληρώσει τα στοιχεία του πατάει το κουμπί **«Εγγραφή»** και η εφαρμογή τον ανακατευθύνει στην σελίδα σύνδεσης.



Μετά την σύνδεση ανοίγει η **κύρια σελίδα της εφαρμογής**.

Κάθε ενότητα μαθημάτων, αποτελεί ένα κουμπί στο **aside** area της σελίδας και ο χρήστης επιλέγει από το **μενού** σε ποια από όλες αυτές θέλει να κατευθυνθεί για να περιηγηθεί στην εφαρμογή. Όταν ο χρήστης κάνει την επιλογή του και πατήσει ένα από τα κουμπιά, θα εμφανιστούν οι αντίστοιχες πληροφορίες τις ενότητας στο **main** area.

Μετά την πρόσβαση του χρήστη στην εφαρμογή, λοιπόν, απεικονίζεται το μενού στο aside area ενώ στο main area οι αρχικές πληροφορίες που δίνονται είναι τα γενικά δεδομένα που αφορούν την C (όπως για παράδειγμα τι είναι οι δεσμευμένες λέξεις στον κώδικα).



Οι πληροφορίες για κάθε ενότητα περιέχουν την **θεωρητική προσέγγιση** της, την παράθεση **κώδικα**, **παραδείγματα**, ρητορικές **ερωτήσεις** για την αφύπνιση της περιέργειας του χρήστη, **τεστ και κουίζ ερωτήσεων** πολλαπλών επιλογών και συμπλήρωσης κενού, **ασκήσεις** για επίλυση, παράθεση έτοιμων **βίντεο**, **άρθρα**, καθώς και την **δημιουργία** μικρών και μεγάλων **προγραμμάτων** βήμα-βήμα.

Επιπλέον, τους παρέχει την δυνατότητα να μπορούν να ανατρέξουν σε οποιαδήποτε ενότητα της εφαρμογής χωρίς να χρειαστεί ένας χρήστης να «ξεκλειδώσει» μια ενότητα κάθε φορά προσφέροντας του την αίσθηση των δια ζώσης μαθημάτων που γίνονται σε πραγματικό χρόνο και έτσι κάνει την εφαρμογή πιο προσιτή για ψάξιμο οποιασδήποτε πληροφορίας, όπως όταν ρωτάς τον δάσκαλο μια ερώτηση την ώρα τον μαθημάτων. Η ιστοσελίδα μας επομένως, προσφέρει μια ολοκληρωμένη εμπειρία εκμάθησης της προγραμματιστικής γλώσσας C και ως ένα full course αλλά και ως μια εφαρμογή εύρεσης πληροφοριών.

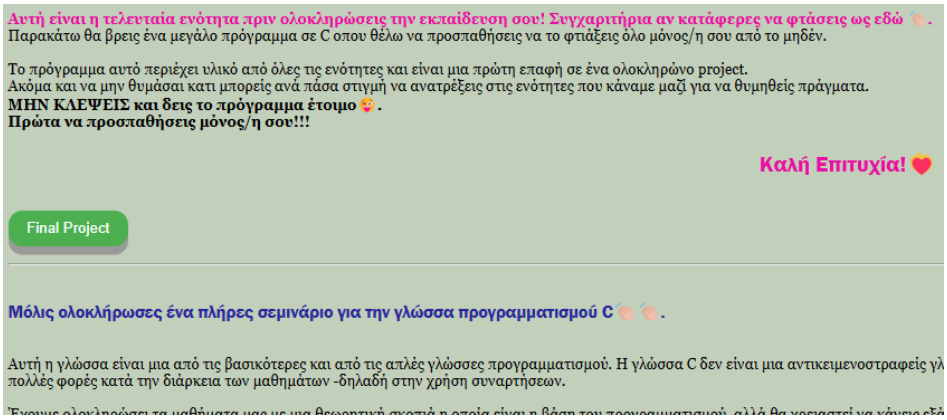
Στο aside area πέρα από το μενού των ενοτήτων, υπάρχουν **τρία** επιπλέον κουμπιά.

- Το πρώτο ονομάζεται **«Βαθμολογία»** και αφορά την συνολική επίδοση του χρήστη που είναι συνδεδεμένος εκείνη την στιγμή. **Η επίδοση σχετίζεται με την βαθμολογία που πήρε σε κάθε κουίζ των ενοτήτων**. Όλες οι ενότητες δεν έχουν τεστ. Κάποιες

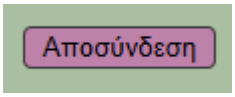
περιλαμβάνουν ολόκληρες ασκήσεις επίλυσης βήμα-βήμα και άλλες περιέχουν ερωτήσεις πολλαπλών επιλογών ή συμπλήρωσης. Οι ενότητες με τα μεγάλα προγράμματα δεν βαθμολογούνται αλλά υπάρχουν για την καλύτερη κατανόηση του θεωρητικού μέρος των μαθημάτων σε μορφή ασκήσεων με εκφωνήσεις που δημιουργήσαμε εμείς. Υπάρχει σταδιακή επίλυση των ασκήσεων και στο τέλος παρατίθενται και ο ολοκληρωμένος κώδικας. Στις ενότητες που υπάρχουν όμως τεστ, αυτά βαθμολογούνται με έναν βαθμό για κάθε σωστή απάντηση του χρήστη. Οι απαντήσεις αποθηκεύονται, επεξεργάζονται για το εάν είναι σωστές ή λανθασμένες και τα αποτελέσματα για κάθε κουίζ και για κάθε ερώτηση εμφανίζονται στην ενότητα «Βαθμολογία». Στο σύνολο υπάρχουν **38 ερωτήσεις** σε όλη την εφαρμογή.

Η ιστοσελίδα είναι δυναμική και αυτό προσφέρει την δυνατότητα η συγκεκριμένη κατηγορία να διαφοροποιείται για κάθε χρήστη, παίρνοντας τα δεδομένα του από την βάση δεδομένων και έτσι του παραθέτει το συνολικό σκορ που κατάφερε να συγκεντρώσει κατά την διάρκεια των μαθημάτων.

- Το δεύτερο κουμπί ονομάζεται **«Τελικό Project σε C»**. Αφορά μια τελική εκφώνηση **άσκησης** σχετικά με όλα όσα έχει διδαχθεί ο χρήστης σε όλες τις διαφορετικές ενότητες. Έχουμε δομήσει την άσκηση να είναι όσο το δυνατόν πιο κατανοητή στον χρήστη, χωρίς να είναι ιδιαίτερα μεγάλης δυσκολίας αλλά να εστιάζει **σε όλη την ύλη** που έχει μάθει, εμβαθύνοντας έτσι στην πιο ουσιαστική εκμάθηση της γλώσσας. Το κουμπί όταν πατηθεί παραθέτει πληροφορίες σχετικές με το project προετοιμάζοντας τον χρήστη για αυτό και τον αποχαιρετάει από την εφαρμογή, αφού είναι η τελευταία ενότητα της ιστοσελίδας και τα μαθήματα έχουν ολοκληρωθεί. Εκεί υπάρχει ένα κουμπί με όνομα **«Final Project»** το οποίο τον κατευθύνει σε μια νέα καρτέλα με την εκφώνηση της άσκησης.



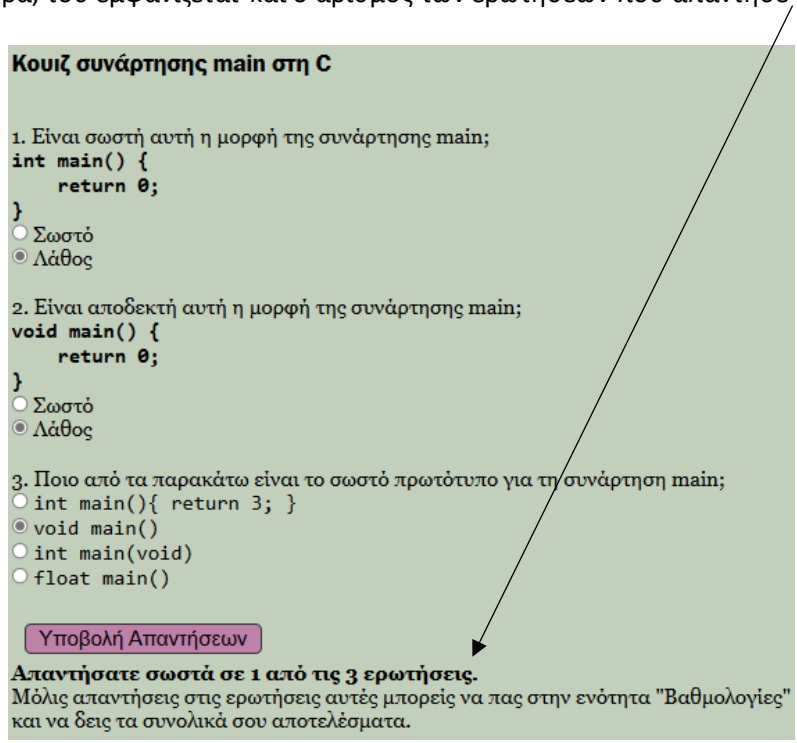
- Το τελευταίο κουμπί που υπάρχει στο aside area λέγεται **«Αποσύνδεση»** και όπως φαίνεται από το όνομα του, αποσκοπεί στην αποσύνδεση του χρήστη από την εφαρμογή, κατευθύνοντας τον ξανά στην σελίδα σύνδεσης.



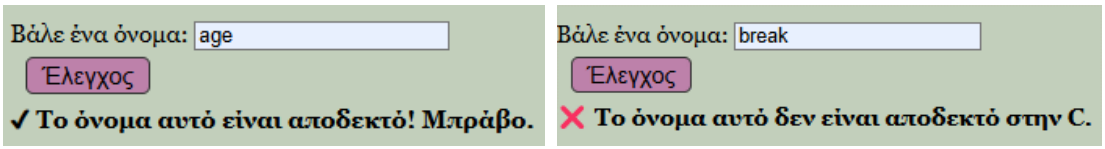
Πάμε, όμως, να δούμε μερικές από τις ενότητες των μαθημάτων λίγο πιο αναλυτικά!

Όπως θα διακρίνουμε το **ύφος της εφαρμογής** είναι παιχνιδιάρικο και ζωηρό, δεδομένου του κοινού στο οποίο απευθύνεται. Για να είναι πιο προσιτό επομένως σε νέους προγραμματιστές και σε μαθητές που βρίσκονται στην δευτεροβάθμια εκπαίδευση, το **λεξιλόγιο** αποτελείται από πιο απλοϊκές λέξεις και εκφράσεις, όπως επίσης και **emoji**. Ταυτόχρονα τα έντονα **χρώματα**, αλλά παράλληλα και ήρεμα στο μάτι του χρήστη που ξυπνούν την περιέργεια και δεν προσδίδει στην εφαρμογή την αίσθηση της μονοτονίας.

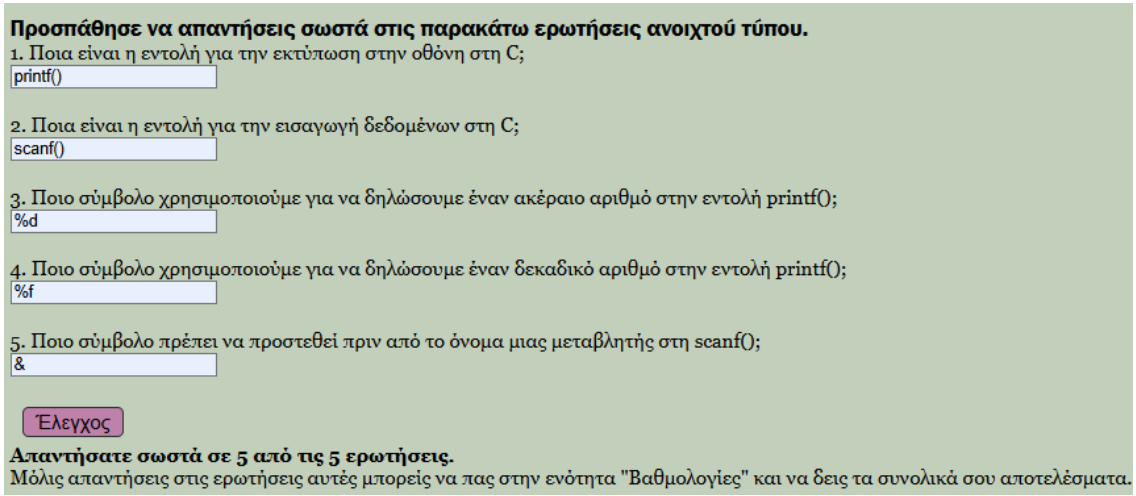
Η πρώτη ενότητα: **«Συνάρτηση main»** είναι η εισαγωγική και αναλύει την κύρια συνάρτηση όλων των προγραμμάτων σε C, την main. Δίνει παραδείγματα κώδικα, αναλύει την σύνταξη της, κάνει μια ελαφριά εισαγωγή στους τύπους δεδομένων, παραθέτει και αναλύει τι είναι μια συνάρτηση και τι είναι ο δομημένος προγραμματισμός στον οποίο στηρίζεται η γλώσσα, πως λειτουργεί η εντολή return, τους διάφορους τύπου όπου μπορούμε να συναρτήσουμε την κύρια συνάρτηση και τέλος πως τοποθετούμε σχόλια στον κώδικα**. Μετά την παράθεση όλων αυτών των πληροφοριών σειρά έχει το πρώτο τεστ της εφαρμογής**. Περιλαμβάνει μια σειρά από τρείς ερωτήσεις πολλαπλής επιλογής και ένα κουμπί υποβολής των απαντήσεων του για να σταλθούν τα δεδομένα με την σειρά τους στον server και στην βάση δεδομένων. Αφού συμπληρωθεί έστω και ένα τεστ **ο χρήστης μπορεί να δει τις απαντήσεις** που έδωσε και τις σωστές απαντήσεις του τεστ στην κατηγορία **«Βαθμολογία»,** ακόμη και μετά τη αποσύνδεση του από την πλατφόρμα. Παρόλα αυτά, εκείνη την ώρα, του εμφανίζεται και ο αριθμός των ερωτήσεων που απάντησε σωστά.



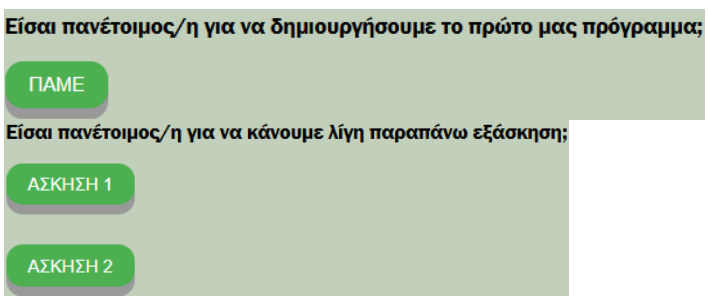
Στην ενότητα **«Μεταβλητές και Σταθερές»** αναφέρονται οι βασικοί τύποι δεδομένων των μεταβλητών και των σταθερών, όπως επίσης και όλοι οι συντακτικοί κανόνες τους οποίους πρέπει να ακολουθούν κατά την δήλωση τους. Μια μεταβλητή πρέπει να ακολουθεί συγκεκριμένους συντακτικούς κανόνες για την δήλωση των ονομάτων της. Η ενότητα αυτή δεν περιέχει κάποιο κουίζ ερωτήσεων, αλλά **περιλαμβάνει μια φόρμα για** να συμπληρώσει ο χρήστης **ένα πιθανό όνομα μεταβλητής και ελέγχει εάν αυτό ακολουθεί τους κανόνες ονομάτων της γλώσσας** (για παράδειγμα ελέγχει να μην είναι κάποια δεσμευμένη λέξη). Ανάλογα την είσοδο του χρήστη εμφανίζει το αντίστοιχο μήνυμα στην οθόνη.



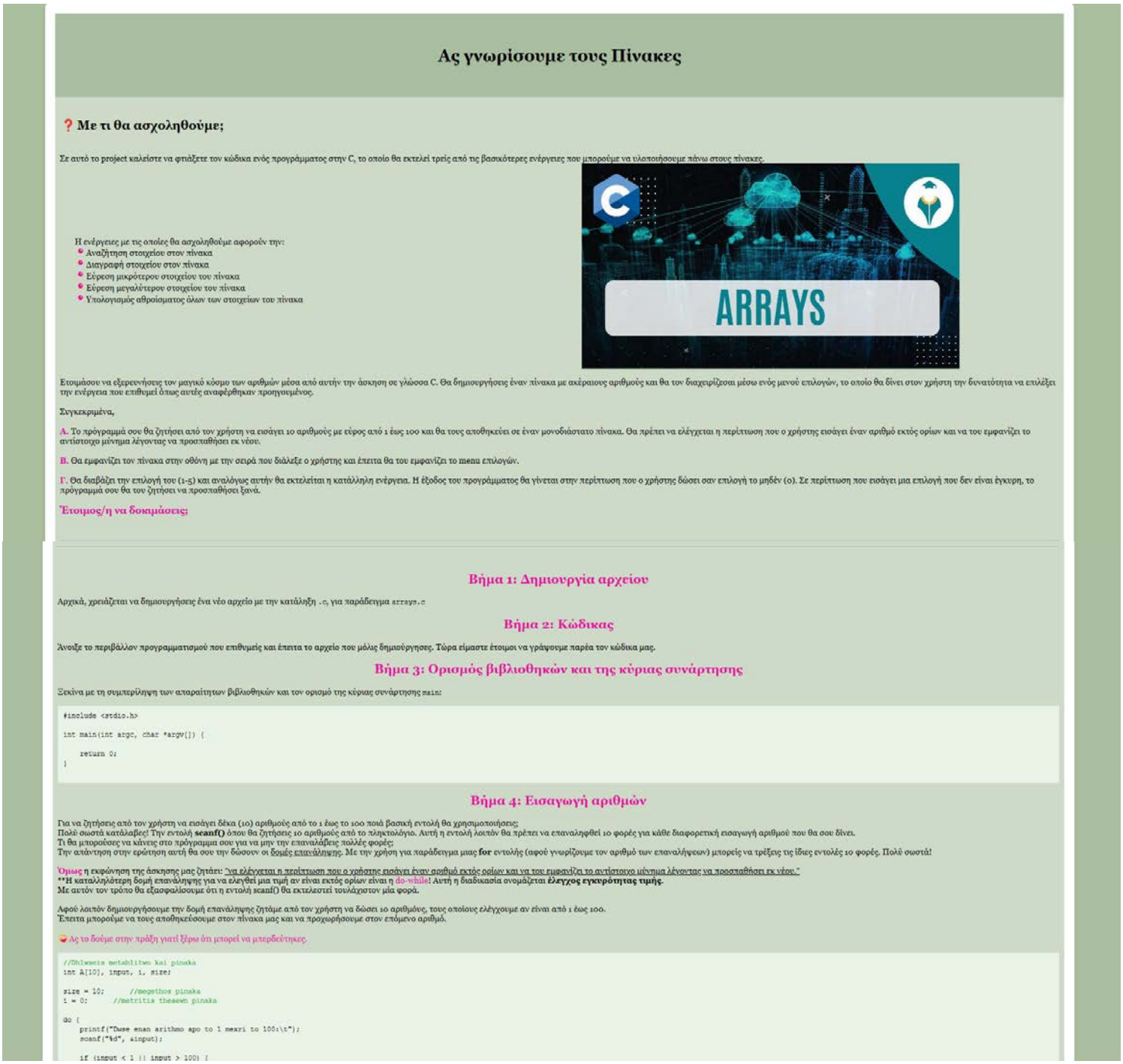
Στην ενότητα **«Βασικές Εντολές και Βιβλιοθήκη stdio.h»** Υπάρχει όλη η θεωρία σχετικά με την βιβλιοθήκη stdio.h και τις δύο βασικές εντολές της, την printf() και την scanf(). Αναλύει πως οι δύο εντολές χρησιμοποιούνται στους διάφορους τύπους δεδομένων, πως λειτουργούν, πως συντάσσονται, δίνει πληροφορίες για τις διευθύνσεις μνήμης και την αποθήκευση των μεταβλητών κ.α. Στο τέλος της ενότητας **υπάρχει** και εδώ ένα **τεστ** ερωτήσεων αλλά **συμπλήρωσης κενών** αυτήν την φορά το οποίο λειτουργεί όπως και το κουίζ ερωτήσεων πολλαπλής επιλογής.



Οι ενότητες: **«Δομές Ελέγχου Ροής (1)»** που αφορά τις **δομές επιλογής** (if, if..else, if..else if…else) και **«Πίνακες»** οι οποίοι χωρίζονται σε **μονής και διπλής διάστασης**. Είναι οι δύο μεγάλες ενότητες των μαθημάτων που δεν περιλαμβάνουν κάποιο τεστ ή κουίζ κατανόησης, αλλά **έχουν τις ασκήσεις των ολοκληρωμένων προγραμμάτων** και της σταδιακής επίλυσης τους. Τα project μεταφέρουν τον χρήστη σε νέα καρτέλα με το πάτημα ενός κουμπιού. Αυτό αποσκοπεί και στην πιο ουσιαστική επαφή του χρήστη με τον προγραμματισμό και την κατανόηση της λειτουργίας του κώδικα.



Οι **σελίδες των ασκήσεων** οπτικά δείχνουν κάπως έτσι:



Ας γνωρίσουμε τους Πίνακες

? Με τι θα ασχοληθούμε;

Σε αυτό το project καλείστε να φτιάξετε τον κώδικα ενός προγράμματος στην C, το οποίο θα εκτελεί τρείς από τις βασικότερες ενέργειες που μπορούμε να υλοποιήσουμε πάνω στους πίνακες.

Η ενέργειες με τις οποίες θα ασχοληθούμε αφορούν την:
- Αναζήτηση στοιχείου στον πίνακα
- Διαγραφή στοιχείου στον πίνακα
- Εύρεση μικρότερου στοιχείου του πίνακα
- Εύρεση μεγαλύτερου στοιχείου του πίνακα
- Υπολογισμός αθροίσματος όλων των στοιχείων του πίνακα

Ετοιμάσου να εξερευνήσεις τον μαγικό κόσμο των αριθμών μέσα από αυτήν την άσκηση σε γλώσσα C. Θα δημιουργήσεις έναν πίνακα με ακέραιους αριθμούς και θα τον διαχειρίζεσαι μέσω ενός μενού επιλογών, το οποίο θα δίνει στον χρήστη την δυνατότητα να επιλέξει την ενέργεια που επιθυμεί όπως αυτές αναφέρθηκαν προηγουμένος.

Συγκεκριμένα,

A. Το πρόγραμμά σου θα ζητήσει από τον χρήστη να εισάγει 10 αριθμούς με εύρος από 1 έως 100 και θα τους αποθηκεύει σε έναν μονοδιάστατο πίνακα. Θα πρέπει να ελέγχεται η περίπτωση που ο χρήστης εισάγει έναν αριθμό εκτός ορίων και να του εμφανίζει το αντίστοιχο μήνυμα λέγοντας να προσπαθήσει εκ νέου.

B. Θα εμφανίζει τον πίνακα στην οθόνη με την σειρά που διάλεξε ο χρήστης και έπειτα θα του εμφανίζει το menu επιλογών.

Γ. Θα διαβάζει την επιλογή του (1-5) και αναλόγως αυτήν θα εκτελείται η κατάλληλη ενέργεια. Η έξοδος του προγράμματος θα γίνεται στην περίπτωση που ο χρήστης δώσει σαν επιλογή το μηδέν (0). Σε περίπτωση που εισάγει μια επιλογή που δεν είναι έγκυρη, το πρόγραμμά σου θα του ζητήσει να προσπαθήσει ξανά.

Έτοιμος/η να δοκιμάσεις;

Βήμα 1: Δημιουργία αρχείου

Αρχικά, χρειάζεται να δημιουργήσεις ένα νέο αρχείο με την κατάληξη `.c`, για παράδειγμα `arrays.c`

Βήμα 2: Κώδικας

Άνοιξε το περιβάλλον προγραμματισμού που επιθυμείς και έπειτα το αρχείο που μόλις δημιούργησες. Τώρα είμαστε έτοιμοι να γράψουμε παρέα τον κώδικα μας.

Βήμα 3: Ορισμός βιβλιοθηκών και της κύριας συνάρτησης

Ξεκίνα με τη συμπερίληψη των απαραίτητων βιβλιοθηκών και τον ορισμό της κύριας συνάρτησης `main`:

```
#include <stdio.h>

int main(int argc, char *argv[]) {

    return 0;
}
```

Βήμα 4: Εισαγωγή αριθμών

Για να ζητήσεις από τον χρήστη να εισάγει δέκα (10) αριθμούς από το 1 έως το 100 ποιά βασική εντολή θα χρησιμοποιήσεις;
Πολύ σωστά κατάλαβες! Την εντολή **scanf()** όπου θα ζητήσεις 10 αριθμούς από το πληκτολόγιο. Αυτή η εντολή λοιπόν θα πρέπει να επαναληφθεί 10 φορές για κάθε διαφορετική εισαγωγή αριθμού που θα σου δίνει.
Τι θα μπορούσες να κάνεις στο πρόγραμμα σου για να μην την επαναλάβεις πολλές φορές;
Την απάντηση στην ερώτηση αυτή θα σου την δώσουν οι δομές επανάληψης. Με την χρήση για παράδειγμα μιας **for** εντολής (αφού γνωρίζουμε τον αριθμό των επαναλήψεων) μπορείς να τρέξεις τις ίδιες εντολές 10 φορές. Πολύ σωστά!

Όμως η εκφώνηση της άσκησης μας ζητάει: "να ελέγχεται η περίπτωση που ο χρήστης εισάγει έναν αριθμό εκτός ορίων και να του εμφανίζει το αντίστοιχο μήνυμα λέγοντας να προσπαθήσει εκ νέου."
****Η καταλληλότερη δομή επανάληψης για να ελεγθεί μια τιμή αν είναι εκτός ορίων είναι η do-while! Αυτή η διαδικασία ονομάζεται **έλεγχος εγκυρότητας τιμής**.
Με αυτόν τον τρόπο θα εξασφαλίσουμε ότι η εντολή scanf() θα εκτελεστεί τουλάχιστον μία φορά.

Αφού λοιπόν δημιουργήσουμε την δομή επανάληψης ζητάμε από τον χρήστη να δώσει 10 αριθμούς, τους οποίους ελέγχουμε αν είναι από 1 έως 100.
Έπειτα μπορούμε να τους αποθηκεύσουμε στον πίνακα μας και να προχωρήσουμε στον επόμενο αριθμό.

Ας το δούμε στην πράξη γιατί ξέρω ότι μπορεί να μπερδεύτηκες.

```
//Dhlwseis metablitwn kai pinaka
int A[10], input, i, size;

size = 10;      //megethos pinaka
i = 0;      //metritis thesewn pinaka

do {
    printf("Dwse enan arithmo apo to 1 mexri to 100:\t");
    scanf("%d", &input);

    if (input < 1 || input > 100) {
```

Το τρίτο διαφορετικό κουίζ ερωτήσεων της ιστοσελίδας βρίσκεται στην ενότητα **«Δομές Ελέγχου Ροής (2)»** , με αυτό να περιλαμβάνει ερωτήσεις πολλαπλής επιλογής με μία σωστή απάντηση, ερωτήσεις πολλαπλής επιλογής με πάνω από μια σωστές απαντήσεις και ερωτήσεις συμπλήρωσης κενού με αριθμό. Η ενότητα σχετίζεται με τις **δομές επανάληψης** που υποστηρίζει η γλώσσα C (for, while, do…while).

Τεστάκι Επανάληψης

1. Ποιο από τα παρακάτω είναι σωστό για την εντολή `for`;
- Εκτελείται τουλάχιστον μία φορά
- Η συνθήκη ελέγχεται μετά από κάθε επανάληψη
- Η συνθήκη ελέγχεται πριν από κάθε επανάληψη
- Δεν χρησιμοποιεί μετρητή

2. Ποια εντολή επανάληψης χρησιμοποιείται όταν **δεν** γνωρίζουμε εκ των προτέρων τον αριθμό επαναλήψεων;
- FOR
- WHILE
- DO...WHILE
- IF

3. Πόσες φορές θα εκτελεστεί ο παρακάτω κώδικας;
```
for (int i = 0; i < 5; i++) { // κώδικας }
```
5

4. Ποιες από τις παρακάτω εντολές είναι δομές επανάληψης;
- [ ] if
- [ ] switch
- [x] for
- [x] while

Υποβολή

Απαντήσατε σωστά σε 4 από τις 4 ερωτήσεις.
Μόλις απαντήσεις στις ερωτήσεις αυτές μπορείς να πας στην ενότητα "Βαθμολογίες" και να δεις τα συνολικά σου αποτελέσματα.

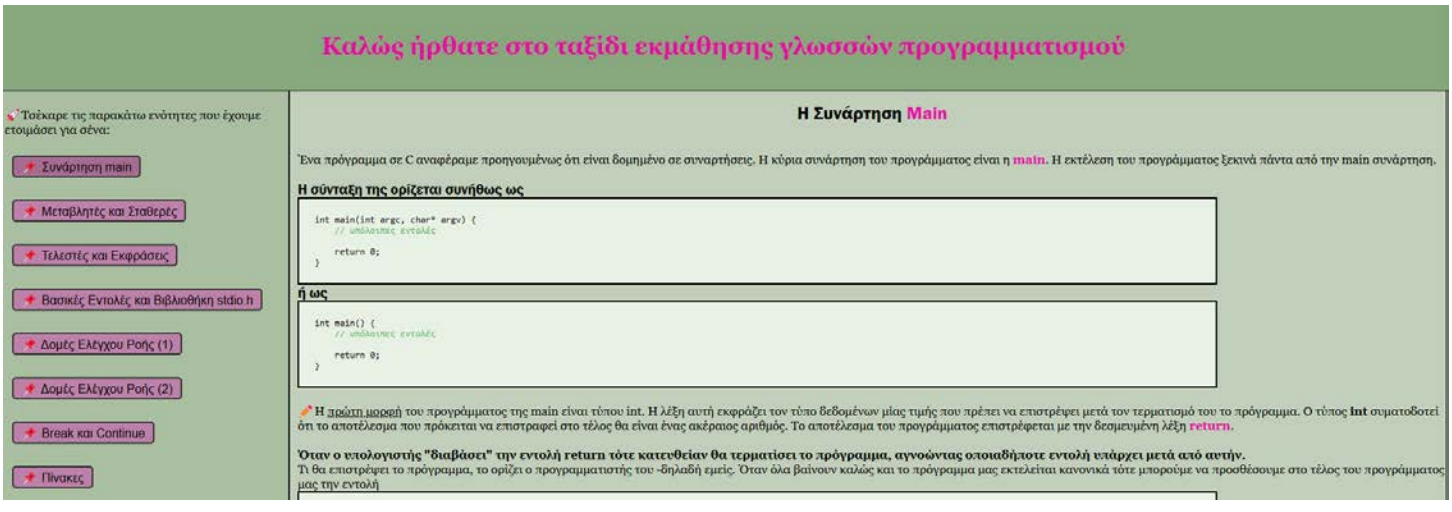
Όλες οι υπόλοιπες ενότητες που υπάρχουν στην πλατφόρμα έχουν κουίζ αυτών των μορφών.

Μετά την ολοκλήρωση την μαθημάτων ο προγραμματιστής θα μπορεί να κατανοήσει, να συγγράψει και να εξηγήσει κώδικα προγραμμάτων γραμμένων στην γλώσσα C.

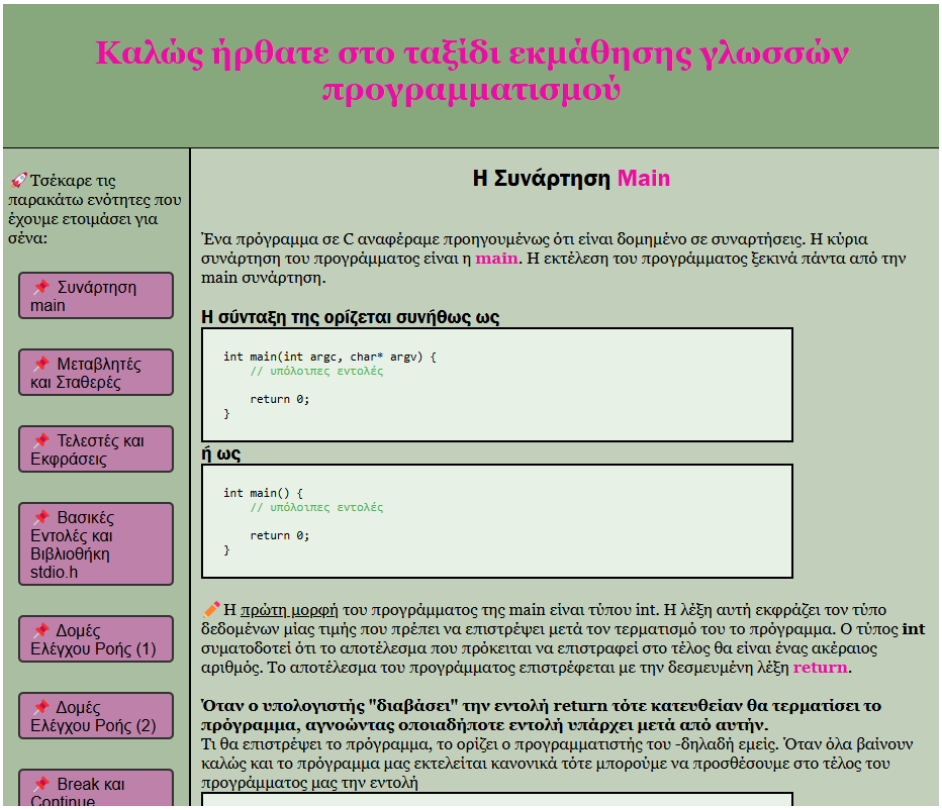
Οι ενότητες, το αναφέραμε νωρίτερα, **δεν εμβαθύνουν** σε ιδιαίτερα δύσκολες έννοιες σχετικές με την γλώσσα όπως τα **αρχεία** (files) , την **δυναμική διαχείριση μνήμης** (όπως δυναμικοί πίνακες) και τις υπόλοιπες **δομές** εκτός των struct (όπως λίστες).

Η πλατφόρμα έχει σχεδιαστεί ώστε να είναι **responsive** σε διαφορετικές συσκευές όπως υπολογιστές, τάμπλετ και κινητά τηλέφωνα.

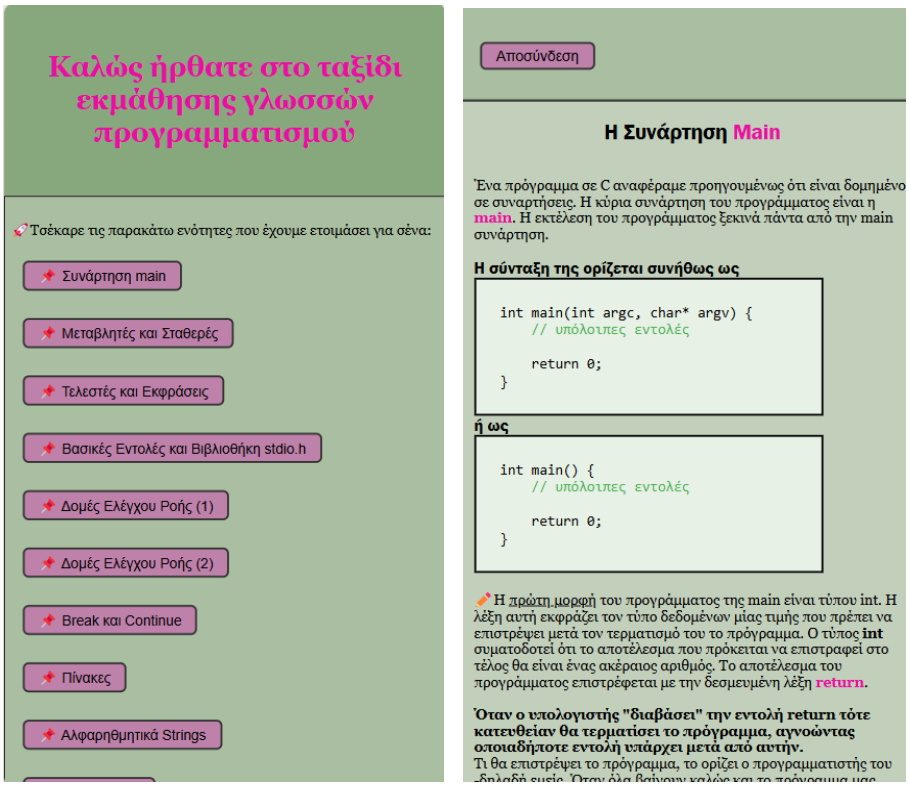
Η συνολική οπτική εικόνα της αρχικής σελίδας σε **Desktop** μοιάζει κάπως έτσι:



Η συνολική οπτική εικόνα της αρχικής σελίδας σε **tablet** μοιάζει κάπως έτσι:



Η συνολική οπτική εικόνα της αρχικής σελίδας σε **smartphone** μοιάζει κάπως έτσι (το aside area με το μενού ενοτήτων μεταφέρεται πάνω από το main area όπως είδαμε πριν στο αρχείο css, για να υπάρχει καλύτερη διαμόρφωση χώρου):



ΚΕΦΑΛΑΙΟ 6 Συμπεράσματα

Σκοπός της παρούσας πτυχιακής εργασίας ήταν να παρουσιαστεί η σημασία των νέων τεχνολογιών μάθησης σε νέα παιδιά και αρχάριους προγραμματιστές. Πλέον ο 24ο αιώνας είναι ο αιώνας της τεχνολογίας και της εξέλιξης. Συνεχώς νέοι πόροι και εργαλεία έρχονται στην επιφάνεια και εμείς καλούμαστε να τους εντάξουμε στην ζωή μας. Η ασύγχρονη εκπαίδευση είναι γεγονός. Ο καθένας μας έχει την καθημερινότητα του και είναι πολύ σημαντικό να μπορεί να προσαρμόσει τις υποχρεώσεις του στο πρόγραμμα του και γενικότερα στους δικούς του ρυθμούς. Για αυτό τον σκοπό το διαδίκτυο έχει ενταχθεί στην ζωή όλων μας.

Όταν λοιπόν ένας άνθρωπος θέλει να ασχοληθεί με τον κόσμο του προγραμματισμού, αντί να χρειαστεί να κάνει διά ζώσης μαθήματα πλέον μπορεί να επιλέξει να μάθει μόνος του από την άνεση του σπιτιού του μια γλώσσα προγραμματισμού. Η εκμάθηση γλωσσών επιτυγχάνεται με διάφορα μέσα τα οποία αναλύθηκαν στα κεφάλαια της εργασίας. Εμείς επιλέξαμε την ανάπτυξη μιας διαδικτυακής πλατφόρμας εκμάθησης γλωσσών προγραμματισμού, επικεντρώνοντας στην γλώσσα προγραμματισμού C, την οποία και παρουσιάσαμε στο προηγούμενο κεφάλαιο. Επιπλέον, για την ανάπτυξη ενός τέτοιου project μπορεί ένας προγραμματιστής να χρησιμοποιήσει διαφορετικές τεχνολογίες ανάλογα με τις απαιτήσεις και τις γνώσεις του πάνω σε αυτές. Είδαμε εκτενώς όλες τις πτυχές για την ανάπτυξη εφαρμογών και δώσαμε στους χρήστες την δυνατότητα μάθησης και εμπειρίας του προγραμματισμού.

Ακόμη, η εφαρμογή αντί να χρησιμοποιηθεί ως εργαλείο μαθημάτων, δίνει την δυνατότητα στον χρήστη να την λειτουργήσει και ως μέσο αναζήτησης πληροφοριών της γλώσσας C και των εννοιών της. Ο χρήστης δεν είναι αναγκασμένος να παρακολουθήσει με την σειρά τα κεφάλαια, αλλά μπορεί να ανατρέξει σε αυτό που επιθυμεί ανά πάσα ώρα. Αυτό αποτελεί ένα ακόμη πλεονέκτημα της εφαρμογής μας.

Γενικότερα, ο κάθε ένας από εμάς, σήμερα έχει την ευκαιρία να διαλέξει το μέσο μάθησης που τον ενδιαφέρει και του ταιριάζει καλύτερα σε σχέση με τα υπόλοιπα, μέσα από μια τεράστια για τα δεδομένα της εποχής πληθώρα επιλογών. Η ασύγχρονη εκπαίδευση προσφέρει στους μαθητές και στις μαθήτριες αυτονομία και ανεξαρτησία στην μάθηση και έτσι ενθαρρύνει την ανάπτυξη κριτικής σκέψης και την δια βίου εκπαίδευση, ιδιαίτερα σε επαγγέλματα που απαιτούν συνεχή ενημέρωση, όπως η πληροφορική. Η συνεχής εξέλιξη των τεχνολογιών μάθησης είναι απαραίτητη για να παραμένουν οι νέοι προγραμματιστές ανταγωνιστικοί. Ενώ η παραδοσιακή μάθηση προσφέρει αλληλεπίδραση με εκπαιδευτικούς και συμμαθητές σε πραγματικό χρόνο, η διαδικτυακή μάθηση προσφέρει την ευελιξία που απαιτείται στις σύγχρονες κοινωνίες, εξοικονομώντας χρόνο και προσαρμόζοντας το περιεχόμενο στις προσωπικές ανάγκες του μαθητή.

Συμπερασματικά, η ανάπτυξη της διαδικτυακής πλατφόρμας εκμάθησης της γλώσσας C αποτέλεσε μια επιτυχημένη απόπειρα προσέγγισης των αρχάριων προγραμματιστών και ειδικά των μαθητών της Δευτεροβάθμιας Εκπαίδευσης. Η πλατφόρμα επιτρέπει στους χρήστες να εισαχθούν στις βασικές αρχές του προγραμματισμού με έναν απλό και κατανοητό τρόπο, διατηρώντας παράλληλα την ευελιξία μάθησης. Ο στόχος της δημιουργίας ενός εύχρηστου και λειτουργικού περιβάλλοντος επιτεύχθηκε, προσφέροντας μια θετική πρώτη επαφή με την προγραμματιστική σκέψη. Ωστόσο, αναγνωρίζεται ότι το ευρύ φάσμα των δυνατοτήτων της γλώσσας C δεν καλύφθηκε πλήρως στην παρούσα «έκδοση» της πλατφόρμας και στο μέλλον μια επέκταση της εφαρμογής με πιο προχωρημένες ενότητες ή επιπλέον γλώσσες προγραμματισμού θα μπορούσε να εμπλουτίσει την εμπειρία του χρήστη.

Ευχαριστώ!

ΒΙΒΛΙΟΓΡΑΦΙΑ

«Η γλώσσα προγραμματισμού C» , Αθανάσιος Κουτσονικόλας , 2021

«Τεχνολογίες και Προγραμματισμός στον Παγκόσμιο Ιστό» , Δουλγέρης, Μαυροπόδι, Κοπανάκι, Καραλής , 2η έκδοση , 2021

«Η γλώσσα C σε βάθος» , Χατζηγιαννάκης , 5η έκδοση

Tillmann, N., Moskal, M., De Halleux, J., Fahndrich, M., Bishop, J., Samuel, A., & Xie, T. (2012, July). The future of teaching programming is on mobile devices. In Proceedings of the 17th ACM annual conference on Innovation and technology in computer science education (pp. 156-161).

Salama, R., Uzunboylu, H., & Alkaddah, B. (2020). Distance learning system, learning programming languages by using mobile applications. New Trends and Issues Proceedings on Humanities and Social Sciences, 7(2), 23-47.

Sharov, S., Tereshchuk, S., Tereshchuk, A., Kolmakova, V., & Yankova, N. (2023). Using MOOC to Learn the Python Programming Language. International Journal of Emerging Technologies in Learning (Online), 18(2), 17.

Abdullah, H. M., & Zeki, A. M. (2014, December). Frontend and backend web technologies in social networking sites: Facebook as an example. In 2014 3rd international conference on advanced computer science applications and technologies (pp. 85-89). IEEE.

Quvvatov, B. (2024). WEB FRONT-END AND BACK-END TECHNOLOGIES IN PROGRAMMING. Theoretical aspects in the formation of pedagogical sciences, 3(1), 208-215.

Sharma, M. (2022). Full Stack Development with MongoDB: Covers Backend, Frontend, APIs, and Mobile App Development Using PHP, NodeJS, ExpressJS, Python and React Native. BPB Publications.

Williams, B., Damstra, D., & Stern, H. (2015). Professional WordPress: design and development. John Wiley & Sons.

Mohamed, M. A., Altrafi, O. G., & Ismail, M. O. (2014). Relational vs. nosql databases: A survey. *International Journal of Computer and Information Technology*, *3*(03), 598-601.

Σερέτης, Δ. Α., & Αργυρόπουλος, Σ. Α. Μ. (2014). Βασικές γνώσεις και εκπαίδευση στο προγραμματισμό με C.

Παπανικολάου, Κ., Γρηγοριάδου, Μ., & Γλέζου, Κ. (2006). Αναπτύσσοντας αλληλεπιδραστικό εκπαιδευτικό υλικό για την εισαγωγή αρχάριων προγραμματιστών στις βασικές δομές προγραμματισμού. Πανελλήνιο Συνέδριο «Ψηφιακό Εκπαιδευτικό Υλικό: ζητήματα δημιουργίας, διδακτικής αξιοποίησης και αξιολόγησης», Βόλος, Απρίλιος.

Ηλεκτρονική:

Ενότητα 2 (upatras.gr)

enotita5.pdf (uoa.gr)

Microsoft Word - Compilers_book.doc (auth.gr)

Front End vs Back End Developer: Διαφορές και Ομοιότητες (bigblue.academy)

Υπολογιστές και Αριθμοί.doc (live.com)

Βασικά στοιχεία προγραμματισμού (ebooks.edu.gr)

Εισαγωγή στον Προγραμματισμό (ebooks.edu.gr)

Υπολογιστικό πρόβλημα - Βικιπαίδεια (wikipedia.org)

C (γλώσσα προγραμματισμού) - Βικιπαίδεια (wikipedia.org)

Μεταγλωττιστής (υπολογιστές) - Βικιπαίδεια (wikipedia.org)

HTML - Wikipedia

JavaScript - Wikipedia

PHP - Wikipedia

Node.js - Wikipedia

WordPress - Βικιπαίδεια (wikipedia.org)

Fortran - Βικιπαίδεια (wikipedia.org)

Γλώσσα προγραμματισμού υψηλού επιπέδου - Βικιπαίδεια (wikipedia.org)

Άντα Λάβλεϊς - Βικιπαίδεια (wikipedia.org)

Γλώσσες Υψηλού Επιπέδου ή Χαμηλού; Οι Διαφορές (bigblue.academy)

Οι 10 πιο δημοφιλείς γλώσσες προγραμματισμού | Randstad

Θες να γίνεις Front-End ή Back-End Developer? (workearly.gr)

Flask Tutorial - GeeksforGeeks

Node.js — The Node.js Event emitter (nodejs.org)

Moving from Touch Develop to MakeCode

GetEmoji - Copy & Paste All Emojis From The Emoji Keyboard - No apps required