ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«Επιλογή Τεχνολογικής Στρατηγικής για υλοποίηση Συστήματος εξ΄Αποστάσεως Εκπαίδευσης μέσω της μεθοδολογίας Ανάλυσης & Σχεδιασμού Συστημάτων.»

Του φοιτητή
Φιλοκώστα Χρήστου
Αρ. Μητρώου: 27/2023

Επιβλέπων
Ονοματεπώνυμο Σ. Κασδερίδης
Βαθμίδα Έκτακτο Διδακτικό Προσωπικό

Ημερομηνία 29/6/2025

Τίτλος Δ.Ε.
Επιλογή Τεχνολογικής Στρατηγικής για υλοποίηση Συστήματος εξ΄Αποστάσεως Εκπαίδευσης μέσω της μεθοδολογίας Ανάλυσης & Σχεδιασμού Συστήματων.
Κωδικός Δ.Ε. 24320
Ονοματεπώνυμο φοιτητή Χρήστος Φιλοκώστας
Ονοματεπώνυμο εισηγητή Στάθης Κασδερίδης
Ημερομηνία ανάληψης Δ.Ε. 01/01/2025
Ημερομηνία περάτωσης Δ.Ε. 15/06/25

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως διπλωματική εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Η έγκριση της διπλωματικής εργασίας από το Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου της Ελλάδος, δεν υποδηλώνει απαραιτήτως και αποδοχή των απόψεων του συγγραφέα, εκ μέρους του Τμήματος.

Αφιερώνεται

Στον επιβλέποντα καθηγητή μου, για την πολύτιμη καθοδήγηση και την εμπιστοσύνη του,
και στην οικογένειά μου, για τη διαρκή υποστήριξη, την αγάπη και την ενθάρρυνσή τους σε κάθε μου βήμα.

Πρόλογος

Η παρούσα διπλωματική εργασία αποτελεί το επιστέγασμα ενός ενδιαφέροντος που ανέπτυξα σταδιακά κατά τη διάρκεια των σπουδών μου: τη δημιουργία και αξιολόγηση λογισμικών λύσεων με άμεση εφαρμογή στον πραγματικό κόσμο. Η επιλογή του θέματος για την υλοποίηση μιας πλατφόρμας διαδικτυακών μαθημάτων με δύο διαφορετικές τεχνολογικές προσεγγίσεις, βασίστηκε στην επιθυμία μου να συγκρίνω πρακτικά τη λειτουργικότητα, την απόδοση και την ευελιξία που προσφέρουν δύο διαφορετικά αρχιτεκτονικά μοντέλα: το WordPress CMS και την αρχιτεκτονική Microservices.

Μέσα από τη διαδικασία αυτή, είχα την ευκαιρία να εφαρμόσω θεωρητικές γνώσεις, να βελτιώσω τις τεχνικές μου δεξιότητες και να αποκτήσω ουσιαστική εμπειρία τόσο στον σχεδιασμό όσο και στην ανάπτυξη λογισμικού. Επιπλέον, κατανόησα σε βάθος τις προκλήσεις που αντιμετωπίζει ένας προγραμματιστής όταν καλείται να υλοποιήσει ένα πληροφοριακό σύστημα που απευθύνεται σε πραγματικούς χρήστες. Το έργο αυτό ενίσχυσε την αυτοπεποίθησή μου και με προετοίμασε καλύτερα για τον επαγγελματικό στίβο στον χώρο της τεχνολογίας.

Περίληψη

Ο στόχος της εργασίας είναι να χρησιμοποιηθεί η μεθοδολογία System Analysis and Design (SAD) για τον σχεδιασμό μιας υπηρεσίας Εξ' Αποστάσεως Εκπαίδευσης. Μέσω της μεθοδολογίας θα αναλυθούν παράγοντες που επιδρούν στην τελική επιλογή λύσης και θα χρησιμοποιηθούν διαδικασίες που βοηθούν στον καθορισμό απαιτήσεων (π.χ. ανάλυση ανταγωνιστών, προσδιορισμός Minimal Viable Product (MVP), κλπ). Ως συνέπεια ο χρήστης της μεθοδολογίας μπορεί να επεκτείνει την αρχική υλοποίηση σε εμπορική βάση, σε μελλοντικό χρόνο, υλοποιώντας μεγαλύτερο μέρος των απαιτήσεων που θα έχουν προσδιοριστεί.

Η εργασία χρησιμοποιεί την μεθοδολογία SAD για να αξιολογήσει δύο αρχιτεκτονικές λύσεις για τον σχεδιασμό και υλοποίηση μιας υπηρεσίας Εξ' Αποστάσεως Εκπαίδευσης σε επίπεδο MVP απαιτήσεων. Η επιλογή της τελικής λύσης γίνεται στη βάση μιας σειράς κριτηρίων (επιχειρησιακών και τεχνολογικών). Για την ενημερωμένη επιλογή υλοποιεί, με δύο διαφορετικές τεχνολογικές στρατηγικές, ένα πρότυπο της λύσης και με βάση την εμπειρία της υλοποίησης των πιλοτικών έργων προτείνει την βαθμολογία των διαφόρων κριτηρίων από την οποία προκύπτει η προτεινόμενη επιλογή. Το αποτέλεσμα είναι η μάθηση μεθόδων εύρεσης απαιτήσεων, μοντελοποίησης, επιλογής λύσης και υλοποίησης πληροφορικών συστημάτων.

«*Selection of a Technological Strategy for Implementing a Distance Learning System through the Systems Analysis & Design Methodology*»

«Christos Filokostas»

Abstract

The objective of this study is to apply the System Analysis and Design (SAD) methodology to the design of a Distance Learning service. Through this methodology, factors influencing the final solution selection will be analyzed, and processes will be employed to assist in requirements definition (e.g., competitor analysis, identification of a Minimum Viable Product (MVP), etc.). As a result, the user of this methodology can later expand the initial implementation into a commercial solution, realizing a greater portion of the previously defined requirements.

In this thesis we use the Systems Analysis & Design methodology to evaluate two architectural solutions for the design and implementation of a Distance Learning service at the level of MVP (Minimum Viable Product) requirements. The final solution is selected based on a set of business and technological criteria. To support an informed decision, two prototype implementations are developed using different technological strategies. Based on the implementation experience of these pilot projects, the various evaluation criteria are scored, leading to the recommended solution. The outcome is a comprehensive understanding of methods for requirements elicitation, modeling, solution selection, and implementation of information systems.

Περιεχόμενα

Κατάλογος Πινάκων

1. Εισαγωγή

1.1. Το πρόβλημα που επιχειρείται να λυθεί

Η ραγδαία ανάπτυξη της εξ αποστάσεως εκπαίδευσης και η αυξημένη ανάγκη για προσβάσιμο, εξατομικευμένο και διαδραστικό ψηφιακό περιεχόμενο έχουν αναδείξει την αξία των πλατφορμών online learning ως κρίσιμα εργαλεία στην εκπαιδευτική διαδικασία. Παρά την πληθώρα λύσεων που υπάρχουν στην αγορά, εξακολουθούν να υπάρχουν σημαντικές προκλήσεις που αφορούν την ευελιξία υλοποίησης, την επεκτασιμότητα, την προσαρμοστικότητα στις ανάγκες του τελικού χρήστη και κυρίως το κόστος ανάπτυξης και συντήρησης [1].

Πολλές υπάρχουσες πλατφόρμες είναι είτε:

- πλήρως εμπορικές και κλειστές (π.χ. Udemy [2], Coursera [3]), περιορίζοντας τον έλεγχο που μπορεί να έχει ένας οργανισμός στο περιβάλλον εκμάθησης,
- είτε πολύπλοκες, απαιτώντας εξειδικευμένο τεχνικό προσωπικό και σημαντικούς πόρους για την ανάπτυξη και διατήρησή τους (π.χ. Moodle LMS [4]).

Με βάση το παραπάνω πλαίσιο, προκύπτει το εξής βασικό ερώτημα:

Ποια τεχνολογική προσέγγιση είναι περισσότερο κατάλληλη για την ανάπτυξη μιας εύχρηστης, επεκτάσιμης και αποδοτικής πλατφόρμας online μαθημάτων; Η χρήση ενός CMS όπως το WordPress ή η κατασκευή με βάση την αρχιτεκτονική Microservices;

Η παρούσα εργασία στοχεύει να απαντήσει στο ερώτημα αυτό μέσα από συστηματική ανάλυση, υλοποίηση και αξιολόγηση δύο τεχνολογικών λύσεων, με σκοπό να προσδιοριστεί ποια προσέγγιση εξυπηρετεί καλύτερα τις λειτουργικές και ποιοτικές απαιτήσεις ενός εκπαιδευτικού οργανισμού ή ενός ανεξάρτητου παρόχου online μαθημάτων.

1.2. Στόχοι της εργασίας

Η παρούσα διπλωματική εργασία έχει ως κύριο στόχο την ανάλυση, υλοποίηση και αξιολόγηση δύο διαφορετικών προσεγγίσεων για την ανάπτυξη μιας πλατφόρμας διαδικτυακών μαθημάτων. Συγκεκριμένα, εξετάζονται:

- Η χρήση του **WordPress** ως πλατφόρμα διαχείρισης περιεχομένου (CMS), αξιοποιώντας κατάλληλα plugins για την υποστήριξη της εκπαιδευτικής λειτουργίας.
- Η υλοποίηση της ίδιας λειτουργικότητας μέσω αρχιτεκτονικής **Microservices**, με υπηρεσίες που συνεργάζονται για να παρέχουν ολοκληρωμένη εμπειρία χρήστη.

Οι βασικοί επιμέρους στόχοι της εργασίας είναι οι εξής:

1. Κατανόηση και αποτύπωση των απαιτήσεων μιας σύγχρονης πλατφόρμας online εκπαίδευσης (σενάρια χρήσης, ρόλοι, λειτουργικές ανάγκες).

2. Σχεδίαση της γενικής λογικής αρχιτεκτονικής της πλατφόρμας, ανεξάρτητα από την τεχνολογική υλοποίηση, με χρήση διαγραμμάτων ροής δεδομένων (DFD).
3. Πρακτική υλοποίηση της πλατφόρμας με δύο διαφορετικές τεχνολογικές προσεγγίσεις:
    - Με χρήση WordPress και κατάλληλων πρόσθετων.
    - Με αρχιτεκτονική μικροϋπηρεσιών (microservices).
4. Συγκριτική ανάλυση των δύο προσεγγίσεων, με βάση μετρήσιμα κριτήρια αξιολόγησης (KPI), όπως επεκτασιμότητα, κόστος, συντηρησιμότητα, απόδοση κ.α.
5. Πρόταση, για την υιοθέτηση της κατάλληλης τεχνολογικής προσέγγισης σε πραγματικά σενάρια, βάσει των αποτελεσμάτων της αξιολόγησης.

Ο τελικός σκοπός είναι η κατάκτηση μιας σφαιρικής, λειτουργικής και στρατηγικής εικόνας, ώστε να υποστηριχθεί η απόφαση επιλογής τεχνολογίας από φορείς εκπαίδευσης, startups ή ανεξάρτητους δημιουργούς εκπαιδευτικού περιεχομένου.

1.3. Μεθοδολογία προσέγγισης (System Analysis & Design)

Η μεθοδολογία που υιοθετήθηκε στην παρούσα διπλωματική εργασία είναι η **System Analysis and Design (SAD)** [5]**[6]**, δηλαδή η Ανάλυση και Σχεδίαση Συστημάτων. Πρόκειται για μια ολοκληρωμένη, οργανωμένη και επιστημονικά τεκμηριωμένη προσέγγιση που έχει ως σκοπό την ορθή ανάπτυξη πληροφοριακών συστημάτων, καλύπτοντας τόσο την τεχνική όσο και τη λειτουργική τους διάσταση. Η SAD αποτελεί βασικό θεμέλιο της Μηχανικής Λογισμικού (Software Engineering) και έχει καθιερωθεί ως διεθνώς αποδεκτό πλαίσιο για την ανάπτυξη σύνθετων συστημάτων.

Η μεθοδολογία αυτή, ακολουθεί μια σαφώς ορισμένη ακολουθία φάσεων, οι οποίες αντανακλούν τον κύκλο ζωής ενός πληροφοριακού συστήματος (System Development Life Cycle – SDLC). Κάθε φάση επιτελεί ένα συγκεκριμένο ρόλο στην πρόοδο του έργου και λειτουργεί ως θεμέλιο για τις επόμενες. Οι κύριες φάσεις είναι οι εξής:

1. Φάση Σχεδιασμού (Planning)

Η πρώτη φάση εστιάζει στον καθορισμό των βασικών στόχων του έργου και στην αξιολόγηση της σκοπιμότητάς του. Γίνεται μια αρχική ανάλυση των αναγκών της επιχείρησης ή του οργανισμού, προσδιορίζονται τα οφέλη και το κόστος της επένδυσης, και λαμβάνονται αποφάσεις για το αν το έργο θα συνεχιστεί. Στο πλαίσιο της παρούσας εργασίας, αυτή η φάση περιλάμβανε την ανάλυση της υπάρχουσας κατάστασης , στις πλατφόρμες εξ αποστάσεως εκπαίδευσης και την διερεύνηση της πιο αποτελεσματικής στρατηγικής για την δημιουργία μιας τέτοιας πλατφόρμας.

2. Φάση Ανάλυσης (Analysis)

Η φάση της ανάλυσης αποτελεί τον πυρήνα της μεθοδολογίας SAD. Σε αυτή τη φάση, αναλύεται εις βάθος η τρέχουσα κατάσταση, καταγράφονται οι διαδικασίες του οργανισμού και εντοπίζονται οι ανάγκες και οι απαιτήσεις των χρηστών. Οι απαιτήσεις διακρίνονται σε:

- **Λειτουργικές απαιτήσεις**: Τι πρέπει να κάνει το σύστημα. Περιλαμβάνουν συγκεκριμένες υπηρεσίες, ροές δεδομένων, λειτουργίες χρηστών, διαδικασίες διαχείρισης και παρακολούθησης.
- **Μη λειτουργικές απαιτήσεις**: Πώς πρέπει να λειτουργεί το σύστημα. Περιλαμβάνουν ζητήματα απόδοσης, ασφάλειας, επεκτασιμότητας, χρηστικότητας και διαθεσιμότητας.

Για τη συλλογή αυτών των απαιτήσεων χρησιμοποιούνται τεχνικές όπως:

- Συνεντεύξεις με stakeholders
- Ερωτηματολόγια και φόρμες
- Παρατήρηση των χρηστών
- Workshops και focus groups
- Μελέτη υφιστάμενων εγγράφων και διαδικασιών

Η ολοκλήρωση της φάσης αυτής αποδίδει το λεγόμενο **requirements definition document**, το οποίο λειτουργεί ως συμβόλαιο ανάμεσα στους αναλυτές και στους τελικούς χρήστες.

3. Φάση Σχεδίασης (Design)

Η φάση σχεδίασης αφορά την τεχνική απεικόνιση του συστήματος. Περιλαμβάνει τον καθορισμό της αρχιτεκτονικής, της βάσης δεδομένων, της δομής διεπαφής χρήστη (UI/UX), των μηχανισμών διαχείρισης και ασφάλειας, καθώς και των αλληλεπιδράσεων μεταξύ συστατικών στοιχείων. Καθορίζονται επίσης τα υποσυστήματα και τα modules του λογισμικού και πώς αυτά θα συνεργάζονται. Η σχεδίαση συχνά εκφράζεται μέσω διαγραμμάτων UML (π.χ., use case diagrams, class diagrams, activity diagrams), τα οποία επιτρέπουν την καθαρή απεικόνιση της λογικής δομής.

4. Φάση Υλοποίησης (Implementation)

Κατά την υλοποίηση, πραγματοποιείται η πραγματική ανάπτυξη του λογισμικού, βάσει του σχεδιασμού που προηγήθηκε. Εδώ περιλαμβάνεται η συγγραφή του κώδικα, η δημιουργία της βάσης δεδομένων, η ενοποίηση των υποσυστημάτων και η διεξαγωγή ενδελεχών δοκιμών. Ένα ακόμα κρίσιμο στάδιο είναι η **δοκιμαστική λειτουργία (testing)**, όπου το σύστημα ελέγχεται για λάθη, λειτουργικότητα, σταθερότητα και ασφάλεια, πριν την οριστική του εγκατάσταση στο παραγωγικό περιβάλλον.

5. Φάση Συντήρησης (Maintenance)

Μετά την παράδοση του συστήματος, αρχίζει η περίοδος συντήρησης. Αυτή περιλαμβάνει διορθώσεις σφαλμάτων, βελτιώσεις στη λειτουργικότητα, προσαρμογές σε νέες απαιτήσεις ή αλλαγές στον εξωτερικό περιβάλλοντα χώρο (τεχνολογικό, νομικό, οργανωτικό). Η συντήρηση αποτελεί μακροχρόνια δραστηριότητα, η οποία διασφαλίζει ότι το σύστημα παραμένει χρήσιμο και λειτουργικό.

Οι **ρόλοι αναλυτών** που ξεχωρίζουν στην μεθοδολογία SAD είναι οι εξής:

Ο **System Analyst** (Αναλυτής Συστημάτων) αποτελεί τον βασικό τεχνικό και συντονιστικό κρίκο ανάμεσα στις ανάγκες του οργανισμού και την τεχνολογική λύση που θα υλοποιηθεί. Ο ρόλος του επικεντρώνεται στην κατανόηση των απαιτήσεων των χρηστών, στη μετάφρασή τους σε τεχνικές προδιαγραφές, και στη συνεργασία με την ομάδα ανάπτυξης για την υλοποίηση του συστήματος. Ο System Analyst συντάσσει τεχνικά έγγραφα, σχεδιάζει αρχιτεκτονικές λύσεις, και συμμετέχει ενεργά στη φάση του testing, διασφαλίζοντας ότι το τελικό προϊόν πληροί τις προδιαγραφές. Επιπλέον, λειτουργεί ως διαμεσολαβητής ανάμεσα στον πελάτη και τους developers, διαχειρίζεται τις αλλαγές στις απαιτήσεις και φροντίζει για τη συμβατότητα της λύσης με τα συστήματα που ήδη χρησιμοποιεί ο οργανισμός.

Ο **Business Analyst** (Αναλυτής Επιχειρησιακών Απαιτήσεων) επικεντρώνεται στην κατανόηση της ευρύτερης επιχειρησιακής στρατηγικής και των λειτουργικών αναγκών ενός οργανισμού. Ο ρόλος του είναι περισσότερο προσανατολισμένος στη διαδικασία και λιγότερο στην τεχνολογία. Ο Business Analyst αναλύει τις επιχειρησιακές ροές, εντοπίζει προβλήματα και ευκαιρίες για βελτιστοποίηση, και προτείνει λύσεις που μπορούν να υποστηριχθούν από τεχνολογικά συστήματα. Παίζει κρίσιμο ρόλο στη φάση της ανάλυσης απαιτήσεων και συχνά συμμετέχει στον καθορισμό των δεικτών επιτυχίας ενός έργου. Η συνεργασία του με τον System Analyst είναι καθοριστική, καθώς μαζί διαμορφώνουν μια τεχνικά υλοποιήσιμη και επιχειρησιακά ωφέλιμη λύση.

Ιστορική Εξέλιξη της Μεθοδολογίας SAD

Η System Analysis and Design δεν είναι νέα έννοια. Οι πρώτες οργανωμένες μορφές της εμφανίστηκαν κατά τη δεκαετία του 1960, όταν οι πρώτες επιχειρήσεις ξεκίνησαν να χρησιμοποιούν ηλεκτρονικούς υπολογιστές για τη διαχείριση δεδομένων. Εκείνη την περίοδο, κυρίαρχη προσέγγιση ήταν η δομημένη ανάλυση (structured analysis), η οποία βασιζόταν σε ροές δεδομένων, πίνακες αποφάσεων και διαγράμματα ροής.

Κατά τη δεκαετία του 1980, η αυξανόμενη πολυπλοκότητα των συστημάτων και η ανάγκη για επαναχρησιμοποίηση κώδικα οδήγησαν στην ανάπτυξη της αντικειμενοστραφούς ανάλυσης και σχεδίασης (object-oriented analysis and design). Αυτή η προσέγγιση προσέφερε καλύτερη μοντελοποίηση της πραγματικότητας, με τη χρήση αντικειμένων, ιδιοτήτων και μεθόδων. Η υιοθέτηση της γλώσσας UML (Unified Modeling Language) στα μέσα της δεκαετίας του 1990 αποτέλεσε καθοριστική τομή, προσφέροντας ένα ενιαίο μέσο περιγραφής και σχεδίασης πληροφοριακών συστημάτων.

Σήμερα, η SAD χρησιμοποιείται είτε αυτούσια είτε σε συνδυασμό με **ευέλικτες μεθοδολογίες (Agile)**, αναδεικνύοντας τον προσαρμοστικό της χαρακτήρα.

1.3.1 Εφαρμογή στην Παρούσα Εργασία

Στο πλαίσιο της παρούσας διπλωματικής εργασίας, η μεθοδολογία SAD προσέφερε τα εξής πλεονεκτήματα:

- **Δομημένη καταγραφή απαιτήσεων**: Εντοπίστηκαν και αναλύθηκαν οι ανάγκες των χρηστών εκπαιδευτικών συστημάτων, τόσο από τεχνική όσο και από παιδαγωγική σκοπιά.
- **Επιστημονική θεμελίωση επιλογών**: Οι επιλογές αρχιτεκτονικής (CMS vs Microservices), τεχνολογιών και εργαλείων τεκμηριώθηκαν με βάση ανάλυση απαιτήσεων και σχεδίαση στόχων.
- **Προσαρμογή στις αρχές MVP (Minimum Viable Product)**: Επιλέχθηκε η ανάπτυξη μιας ελάχιστης λειτουργικής έκδοσης, ικανής να παρέχει τις βασικές υπηρεσίες, με δυνατότητα μελλοντικής επέκτασης.
- **Τεκμηρίωση με χρήση UML**: Χρησιμοποιήθηκαν διαγράμματα για την περιγραφή ροής χρηστών, λογικής αρχιτεκτονικής, και ροής δεδομένων.

Η επιλογή της συγκεκριμένης μεθοδολογίας συνέβαλε καθοριστικά στην επιτυχία της ανάλυσης και του σχεδιασμού του υπό μελέτη συστήματος, ενώ ταυτόχρονα επέτρεψε την τεκμηριωμένη σύγκριση μεταξύ των τεχνολογικών εναλλακτικών που εξετάζονται.

Η επιλογή της μεθοδολογίας **System Analysis and Design (SAD)** αποτέλεσε βασικό άξονα στην ανάπτυξη του εκπαιδευτικού πληροφοριακού συστήματος που παρουσιάζεται στην παρούσα εργασία. Η SAD όχι μόνο προσέφερε ένα σαφές πλαίσιο οργάνωσης της ανάλυσης και της σχεδίασης, αλλά ταυτόχρονα κατέστησε δυνατή την ορθολογική λήψη αποφάσεων σε κάθε φάση του έργου. Μέσα από την εφαρμογή των αρχών της μεθοδολογίας, επιτεύχθηκε η επίτευξη ενός αποτελέσματος που είναι τεχνικά υλοποιήσιμο, παιδαγωγικά αποτελεσματικό και τεκμηριωμένα επεκτάσιμο.

Δομημένη καταγραφή απαιτήσεων

Η ανάλυση των απαιτήσεων στην παρούσα εργασία πραγματοποιήθηκε αποκλειστικά από τον συγγραφέα, με βάση συνδυασμένη προσέγγιση επιχειρηματικής λογικής, τεχνολογικής σκοπιμότητας και εμπειρικής γνώσης των αναγκών του εκπαιδευτικού περιβάλλοντος. Δεδομένου ότι η εργασία δεν εκπονήθηκε στο πλαίσιο μιας πολυμελούς ομάδας ανάπτυξης με διακριτούς ρόλους, η ανάλυση προσεγγίστηκε από επιχειρηματική σκοπιά: δηλαδή, ποια προστιθέμενη αξία μπορεί να προσφέρει ένα τέτοιο σύστημα σε πραγματικό εκπαιδευτικό ή οργανωτικό περιβάλλον, ποιες λειτουργίες θεωρούνται ουσιώδεις για τους χρήστες, και πώς η τεχνολογία μπορεί να εξυπηρετήσει τους στόχους αυτούς με βιώσιμο τρόπο.

Η τεκμηρίωση των απαιτήσεων βασίστηκε σε μελέτη υφιστάμενων λύσεων, ανάλυση χρήσης εκπαιδευτικών συστημάτων (Learning Management Systems – LMS) και προσομοίωση ρεαλιστικών σεναρίων χρήσης. Παρά την απουσία πρωτογενούς συλλογής δεδομένων (π.χ. μέσω συνεντεύξεων ή focus groups), χρησιμοποιήθηκε η τεχνική της εξαγόμενης απαίτησης: δηλαδή, η συναγωγή των αναγκών των χρηστών με βάση παρατηρήσεις από παρόμοια συστήματα και τις λειτουργικές ελλείψεις τους.

Η συγκεκριμένη προσέγγιση, μολονότι περιορίζεται σε ανάλυση από ένα μόνο άτομο, προσέφερε σαφή στόχευση και λειτουργική αρτιότητα, καθώς βασίστηκε σε πραγματικές και επαναλαμβανόμενες ανάγκες της εκπαιδευτικής πράξης. Το αποτέλεσμα ήταν ένα σύνολο απαιτήσεων που κατευθύνει με ακρίβεια τον σχεδιασμό του πληροφοριακού συστήματος, αποφεύγοντας περιττή πολυπλοκότητα και προβάλλοντας ουσιώδη χαρακτηριστικά.

Τεκμηρίωση επιλογών

Μέσα από τη μεθοδολογία SAD καθορίστηκαν κριτήρια επιλογής τεχνολογιών και αρχιτεκτονικών λύσεων. Ένα από τα πλέον χαρακτηριστικά παραδείγματα είναι η συγκριτική ανάλυση μεταξύ **μονολιθικής αρχιτεκτονικής CMS (π.χ. Moodle)** και **κατανεμημένης αρχιτεκτονικής μικροϋπηρεσιών (microservices)**. Η ανάλυση περιέλαβε ποσοτικά και ποιοτικά κριτήρια όπως: ευκολία ανάπτυξης, δυνατότητα παραμετροποίησης, επεκτασιμότητα, διαλειτουργικότητα, ασφάλεια και συντηρησιμότητα.

Η SAD προσέγγιση επέβαλε τη διατύπωση ρητών τεχνικών προδιαγραφών, όπως η ανάγκη για modular δομή, υποστήριξη RESTful APIs, διαχωρισμό λογικών επιπέδων (presentation, logic, data), και συμβατότητα με υπάρχοντα πρότυπα (π.χ. SCORM για e-learning). Οι τελικές επιλογές βασίζονται στην τεχνική αιτιολόγηση που προέκυψε από την ανάλυση, και όχι σε προσωπικές προτιμήσεις ή συμβατικές λύσεις.

Προσαρμογή στις αρχές Minimum Viable Product (MVP)

Η ανάπτυξη ενός **MVP (Minimum Viable Product)** αποτέλεσε στρατηγική απόφαση, σύμφωνη με την ευέλικτη φιλοσοφία που ενσωματώνει πλέον η σύγχρονη SAD. Η υιοθέτηση MVP σήμαινε την υλοποίηση ενός συστήματος το οποίο παρέχει **τον ελάχιστο απαραίτητο πυρήνα λειτουργικότητας** (core features), ικανό να αποδείξει τη βιωσιμότητα της λύσης και να συγκεντρώσει ανατροφοδότηση από τους χρήστες.

Αυτό περιλάμβανε λειτουργίες όπως:

- δημιουργία και ανάθεση μαθημάτων,
- διαχείριση χρηστών με διαφορετικά επίπεδα πρόσβασης,
- δυνατότητα υποβολής και αξιολόγησης μαθημάτων,
- βασικά στατιστικά προόδου μαθητή.

Η προσέγγιση MVP όχι μόνο επιτάχυνε την αρχική ανάπτυξη, αλλά διατήρησε και την αρχιτεκτονική επεκτάσιμη, επιτρέποντας την προσθήκη πιο σύνθετων χαρακτηριστικών (π.χ. gamification, adaptive learning) σε μελλοντικές φάσεις.

Τεκμηρίωση με χρήση UML

Σύμφωνα με τη SAD και ειδικότερα την αντικειμενοστραφή της εκδοχή, η χρήση της **Unified Modeling Language (UML)** ήταν καθοριστική για τη σχεδίαση του συστήματος. Στην εργασία αξιοποιήθηκαν διαφορετικοί τύποι UML διαγραμμάτων για την απεικόνιση της εσωτερικής και εξωτερικής συμπεριφοράς του συστήματος:

- **Class Diagrams**, για τη σχεδίαση της λογικής δομής του λογισμικού.
- **Activity Diagrams**, για την αποτύπωση της ροής εργασιών (workflows) σε βασικά σενάρια χρήσης.

Η χρήση αυτών των εργαλείων συνέβαλε στην ενίσχυση της κατανόησης του συστήματος, στην αποφυγή ασαφειών, και στη διευκόλυνση της συντήρησης και επαναχρησιμοποίησης του κώδικα.

Γενική αποτίμηση της εφαρμογής SAD

Η μεθοδολογία SAD παρείχε ένα σταθερό και μεθοδικό πλαίσιο που κάλυψε κάθε στάδιο της ανάπτυξης, από τη διερεύνηση των αναγκών μέχρι την τεκμηρίωση των τεχνικών αποφάσεων. Παρείχε τη δυνατότητα σαφούς επικοινωνίας μεταξύ τεχνικών και μη τεχνικών stakeholders, μείωσε τον κίνδυνο αστοχίας και δημιούργησε ισχυρή βάση για μελλοντική επέκταση ή επαναχρησιμοποίηση του συστήματος.

Επιπλέον, λειτούργησε ως πλαίσιο τεκμηρίωσης της σύγκρισης μεταξύ CMS και Microservices, καθώς η συστηματική προσέγγιση της ανάλυσης απαιτήσεων και του σχεδιασμού διευκόλυνε την αξιολόγηση των δύο τεχνολογιών με βάση μετρήσιμα και ρεαλιστικά κριτήρια.

1.4. Δομή της διπλωματικής εργασίας

Η δομή της παρούσας διπλωματικής εργασίας ακολουθεί μια λογική σειρά που ξεκινά από τη θεωρητική τεκμηρίωση του προβλήματος και καταλήγει στην τεχνική υλοποίηση και αξιολόγηση των λύσεων. Στο πρώτο κεφάλαιο παρουσιάζεται το βασικό πρόβλημα, οι στόχοι της εργασίας, η μεθοδολογία που ακολουθείται (System Analysis & Design) καθώς και η συνολική οργάνωση του περιεχομένου. Στο δεύτερο κεφάλαιο γίνεται αναλυτική αποτύπωση των απαιτήσεων της πλατφόρμας διαδικτυακής εκπαίδευσης, περιλαμβάνοντας τη μελέτη ανταγωνιστικών λύσεων, τον καθορισμό του ελάχιστου βιώσιμου προϊόντος (MVP) και τη διαμόρφωση του λογικού μοντέλου λειτουργίας μέσω διαγραμμάτων ροής δεδομένων (DFD). Στη συνέχεια, στο τρίτο κεφάλαιο, παρουσιάζονται αναλυτικά οι τεχνολογικές βάσεις των δύο διαφορετικών προσεγγίσεων: η αξιοποίηση του WordPress ως σύστημα διαχείρισης περιεχομένου (CMS) και η ανάπτυξη της ίδιας λειτουργικότητας μέσω αρχιτεκτονικής Microservices. Ακολουθεί το τέταρτο κεφάλαιο, όπου περιγράφεται η διαδικασία υλοποίησης των δύο λύσεων, με έμφαση στη λειτουργική ροή, τις επιμέρους τεχνολογικές επιλογές και την απεικόνιση των διαδρομών

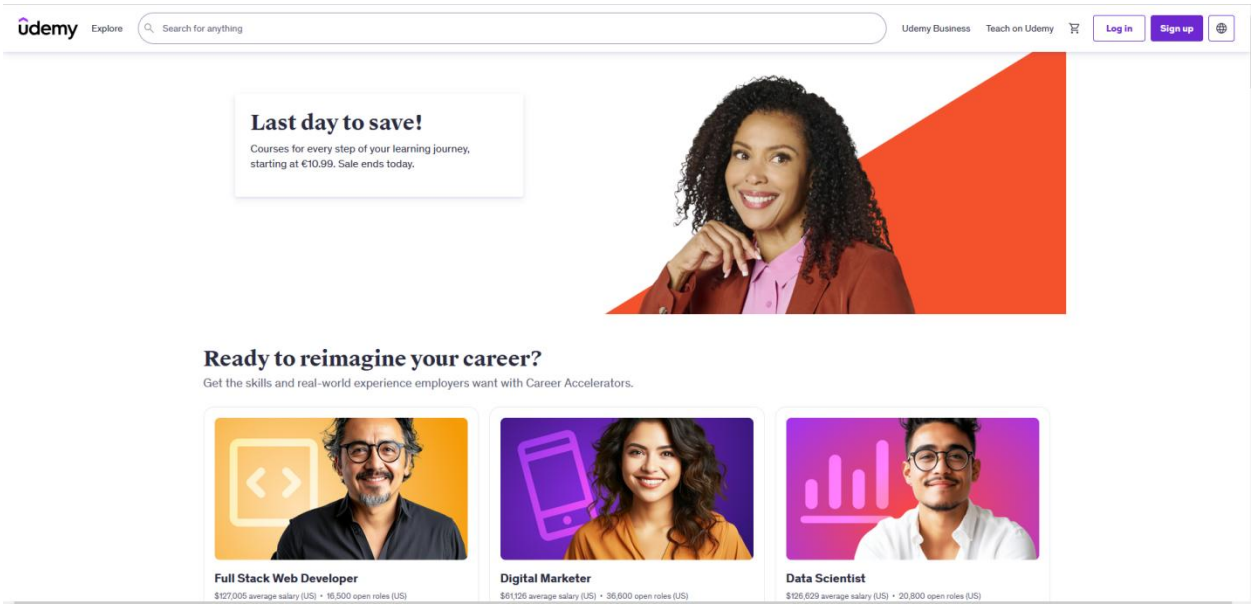
χρήστη μέσω DFD. Το πέμπτο κεφάλαιο επικεντρώνεται στη συγκριτική ανάλυση των δύο προσεγγίσεων, βάσει κοινών λειτουργικών χαρακτηριστικών και με τη βοήθεια KPIs (Key Performance Indicators), ενώ παρουσιάζεται και ένα εικονικό πλάνο για την ανάπτυξη (deployment) της πλατφόρμας στον «πραγματικό κόσμο». Τέλος, στο έκτο κεφάλαιο συνοψίζονται τα βασικά συμπεράσματα της εργασίας, τα τεχνικά και επιχειρησιακά οφέλη που αποκομίστηκαν, καθώς και προτάσεις για μελλοντική εξέλιξη ή εμβάθυνση της παρούσας εργασίας.

2. Ανάλυση Απαιτήσεων και Σχεδίαση Συστήματος

2.1. Ανάλυση ανταγωνιστικών πλατφορμών

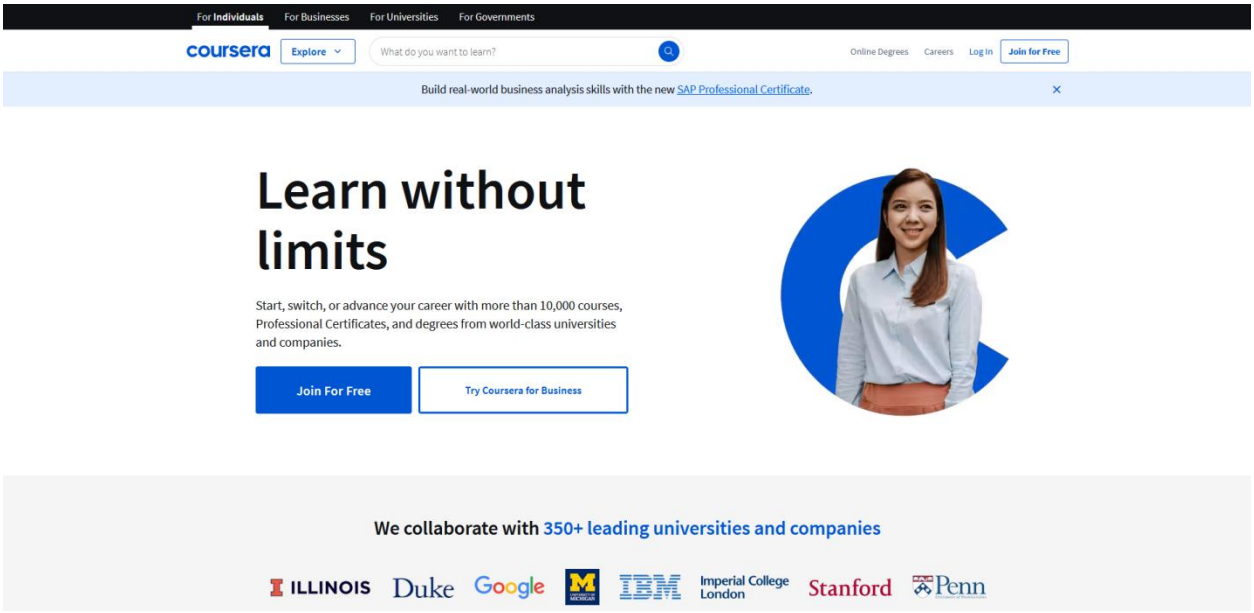
Στο πλαίσιο της παρούσας εργασίας, είναι κρίσιμη η κατανόηση των χαρακτηριστικών και των λειτουργικών δυνατοτήτων που προσφέρουν ήδη καθιερωμένες πλατφόρμες διαδικτυακής εκπαίδευσης, προκειμένου να καθοριστεί ένα ρεαλιστικό και ανταγωνιστικό πρότυπο σχεδίασης. Οι πιο διαδεδομένες πλατφόρμες στην αγορά όπως η **Udemy** [2], το **Coursera** [3] και το **Moodle** [4] αποτελούν σημαντικά σημεία αναφοράς, καθώς προσφέρουν πλούσιο σύνολο δυνατοτήτων, ευχρηστία και μεγάλη εμπειρία χρήστη (UX).

Η **Udemy** είναι μια εμπορική πλατφόρμα που επιτρέπει σε ανεξάρτητους εκπαιδευτές να δημιουργούν και να πωλούν τα δικά τους μαθήματα. Το μοντέλο της βασίζεται στην αυτοματοποίηση της διαδικασίας πώλησης, παρακολούθησης και αξιολόγησης των μαθημάτων. Η εμπειρία χρήστη είναι ιδιαίτερα φιλική, αλλά η πλατφόρμα είναι κλειστή: δεν επιτρέπει σε εξωτερικούς οργανισμούς να παραμετροποιήσουν πλήρως το περιβάλλον, ούτε προσφέρει πρόσβαση στον πηγαίο κώδικα ή δυνατότητες προσαρμοσμένων λειτουργιών εκτός όσων προσφέρει η ίδια.



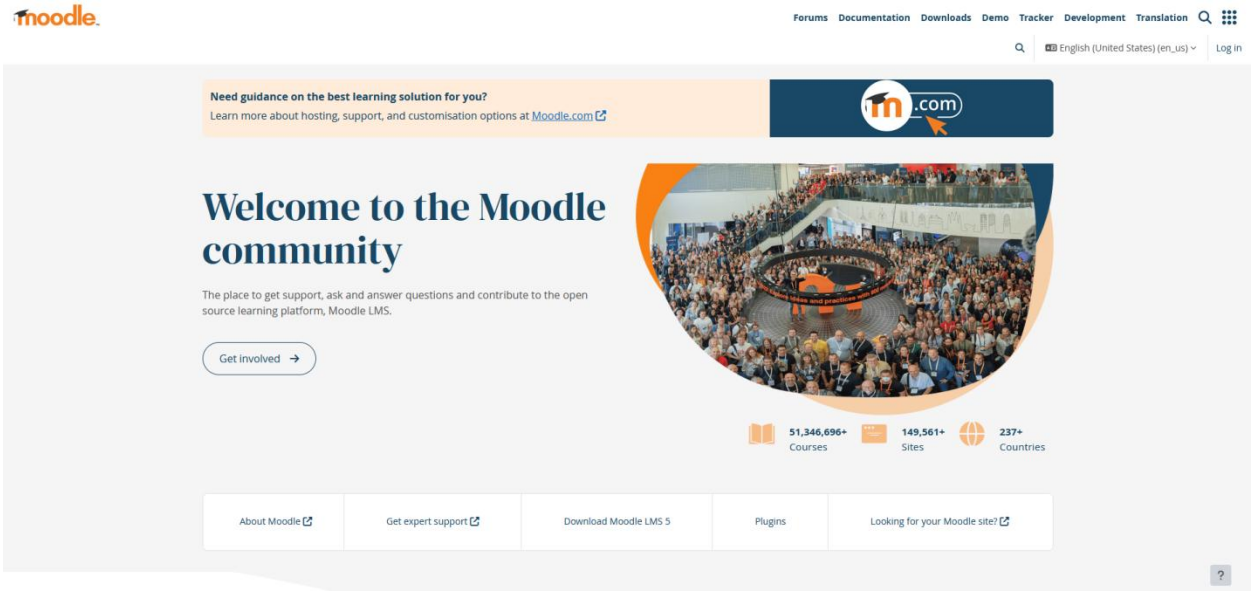
Εικόνα 2.1: Αρχική σελίδα της πλατφόρμας Udemy

Το **Coursera** εστιάζει περισσότερο σε συνεργασίες με πανεπιστήμια και εκπαιδευτικούς φορείς, προσφέροντας μαθήματα υψηλής ακαδημαϊκής ποιότητας. Παρέχει ευρεία υποστήριξη για quizzes, βαθμολογίες, πιστοποιήσεις και καθοδηγούμενη μάθηση, όμως λειτουργεί επίσης ως κλειστό οικοσύστημα, χωρίς δυνατότητα πλήρους ελέγχου από εξωτερικούς παρόχους.



Εικόνα 2.2: Αρχική σελίδα της πλατφόρμας coursera

Αντιθέτως, το **Moodle** είναι μια λύση ανοικτού κώδικα που απευθύνεται κυρίως σε εκπαιδευτικά ιδρύματα. Προσφέρει μεγάλη ευελιξία, δυνατότητα φιλοξενίας σε ιδιόκτητους servers και πλούσια παραμετροποίηση, αλλά συχνά χαρακτηρίζεται από αυξημένη πολυπλοκότητα εγκατάστασης και διαχείρισης, γεγονός που απαιτεί τεχνική εξειδίκευση.



Εικόνα 2.3: Αρχική σελίδα της πλατφόρμας Moodle

Η επισκόπηση των παραπάνω πλατφορμών ανέδειξε τόσο τα ισχυρά σημεία όσο και τους περιορισμούς κάθε προσέγγισης. Συγκεκριμένα, παρατηρείται ότι οι εμπορικές λύσεις υπερτερούν σε εμπειρία χρήστη

και προσβασιμότητα, αλλά υστερούν σε δυνατότητα παραμετροποίησης και τεχνολογική ελευθερία. Αντίστοιχα, οι ανοικτού κώδικα λύσεις προσφέρουν πλήρη έλεγχο αλλά προϋποθέτουν αυξημένο τεχνικό κόστος.

Η σύγκριση αυτή βοήθησε στη διαμόρφωση του λειτουργικού πλαισίου της προτεινόμενης πλατφόρμας και στον καθορισμό του **Minimum Viable Product (MVP)**, όπως θα παρουσιαστεί στο επόμενο κεφάλαιο.

2.2. Καθορισμός MVP (Minimum Viable Product)

Το **Minimum Viable Product (MVP)** αναφέρεται στην πιο απλή έκδοση ενός προϊόντος που μπορεί να παραδοθεί στον τελικό χρήστη, παρέχοντας επαρκή λειτουργικότητα ώστε να είναι χρηστικό και να εξυπηρετεί τον βασικό του σκοπό. Ουσιαστικά, αποτελεί την αρχική υλοποίηση ενός προϊόντος με τις απολύτως απαραίτητες λειτουργίες, ώστε να μπορεί να τεθεί σε χρήση, να δοκιμαστεί από τους χρήστες και να προσφέρει άμεσα αξία, χωρίς να απαιτείται πλήρης και χρονοβόρα ανάπτυξη όλων των χαρακτηριστικών του τελικού προϊόντος [7].

Στόχος του MVP είναι η γρήγορη επιβεβαίωση της χρησιμότητας ενός προϊόντος (product validation), η μείωση του κόστους και του τεχνικού ρίσκου, καθώς και η συλλογή πρώιμης ανατροφοδότησης από τους χρήστες. Η προσέγγιση αυτή είναι ιδιαίτερα σημαντική σε περιβάλλοντα όπου απαιτείται ευελιξία και συνεχής προσαρμογή στις ανάγκες των τελικών χρηστών.

Η Ιστορική Καταγωγή και Διάδοση του MVP

Η έννοια του Minimum Viable Product (MVP), δηλαδή του «ελάχιστα βιώσιμου προϊόντος», εμφανίστηκε για πρώτη φορά το 2001, διατυπωμένη από τον Frank Robinson, συνιδρυτή της εταιρείας SyncDev. Ο Robinson όρισε το MVP ως το προϊόν με τη μικρότερη δυνατή λειτουργικότητα που μπορεί ωστόσο να προσφέρει αξία στον χρήστη και να επικυρώσει βασικές υποθέσεις αγοράς, χωρίς να επιβαρύνει την ανάπτυξη με περιττή πολυπλοκότητα ή κόστος. Αυτή η προσέγγιση βασίζεται στην αρχή ότι η έγκαιρη ανατροφοδότηση από την αγορά είναι πιο χρήσιμη από τη θεωρητική τελειότητα του προϊόντος.

Περίπου δέκα χρόνια αργότερα, ο Eric Ries καθιέρωσε τον όρο MVP σε διεθνές επίπεδο μέσα από το βιβλίο του *The Lean Startup* (2011). Ο Ries ενέταξε το MVP στον ευρύτερο κύκλο ανάπτυξης *Build-Measure-Learn*, όπου κάθε νέο χαρακτηριστικό ή προϊόν αποτελεί ένα πείραμα για επικυρωμένη μάθηση. Χάρη στο έργο του, το MVP αναδείχθηκε ως θεμελιώδης έννοια για τη στρατηγική ανάπτυξη καινοτόμων προϊόντων, τόσο σε νεοφυείς επιχειρήσεις όσο και σε μεγάλους οργανισμούς. [8],

Μεθοδολογική εφαρμογή στην παρούσα εργασία

Η διαδικασία καθορισμού του MVP στην παρούσα εργασία βασίστηκε στις αρχές και στα βήματα που προτείνει ο Eric Ries στο *The Lean Startup*, όπου το προϊόν δεν αντιμετωπίζεται απλώς ως τεχνικό σύνολο λειτουργιών, αλλά ως πείραμα για την επικύρωση επιχειρηματικών ή λειτουργικών υποθέσεων. Βασική προϋπόθεση της μεθοδολογίας είναι ότι κάθε προϊόν ξεκινά από μια ή περισσότερες κρίσιμες υποθέσεις για το τι θέλουν ή χρειάζονται οι χρήστες. Ο μόνος τρόπος να δοκιμαστούν αυτές οι υποθέσεις είναι η επαφή με την πραγματικότητα – και αυτό ακριβώς επιτυγχάνει το MVP.

Σε αυτό το πλαίσιο, ο συγγραφέας, έχοντας τον ρόλο τόσο του σχεδιαστή όσο και του αναλυτή, ξεκίνησε τη διαδικασία από την αρχή της formulate–test–learn, που προτείνει ο Ries μέσω του κύκλου *Build–Measure–Learn*. Το πρώτο βήμα ήταν η διατύπωση των βασικών υποθέσεων χρήσης: ότι δηλαδή οι εκπαιδευτικοί χρειάζονται ένα εργαλείο απλό, λειτουργικό, και εστιασμένο στη βασική διαχείριση μαθημάτων, χωρίς την πολυπλοκότητα ενός πλήρους LMS, και ότι οι μαθητές μπορούν να αλληλεπιδρούν με αυτό αποδοτικά χωρίς καμπύλη εκμάθησης.

Ακολούθως, έγινε καταγραφή των αναγκών με βάση εμπειρικά δεδομένα και συγκριτική μελέτη παρόμοιων συστημάτων. Αντί να υιοθετηθεί μια προσεγγιστική λίστα επιθυμητών λειτουργιών, εφαρμόστηκε η μεθοδολογία “Problem/Solution Fit”. Σύμφωνα με αυτήν, η ανάπτυξη πρέπει να ξεκινά όχι από αυτό που *μπορούμε* να κατασκευάσουμε, αλλά από αυτό που *πρέπει* να επιλύσουμε. Ο στόχος δεν ήταν να παραχθεί μια πλήρης πλατφόρμα, αλλά να επαληθευτεί αν μια βασική έκδοση της πλατφόρμας μπορεί να προσφέρει αξία και να λύσει πραγματικά προβλήματα των χρηστών.

Μέσα από αυτή τη διαδικασία απορρίφθηκαν λειτουργίες που ανήκαν στη σφαίρα του “nice to have” (π.χ. gamification, εξελιγμένα analytics, συστήματα αξιολόγησης peer-to-peer) και κρατήθηκαν μόνο εκείνες που αντιστοιχούν στον ελάχιστο πυρήνα που προσφέρει ουσιαστική εμπειρία χρήσης. Έτσι προέκυψε μια MVP εκδοχή που περιλάμβανε τις βασικές λειτουργίες.

Η τελική επιλογή αυτού του MVP έγινε με βάση δύο παραμέτρους του *Lean Startup*:

1. Ταχύτητα απόκρισης: Το σύστημα έπρεπε να είναι αρκετά απλό ώστε να υλοποιηθεί και να δοκιμαστεί άμεσα.
2. Επικυρωμένη μάθηση (validated learning): Το MVP όφειλε να παράγει αποτελέσματα που να βοηθούν στη λήψη τεκμηριωμένων αποφάσεων για την περαιτέρω ανάπτυξη.

Η στρατηγική αυτή δεν ήταν απλώς τεχνική, αλλά μεθοδολογική: επιλέχθηκε ο δρόμος της επανάληψης, της ταχείας δοκιμής και της προσαρμογής. Αντί να επενδυθεί μεγάλος χρόνος σε ένα πλήρως ανεπτυγμένο σύστημα που ενδεχομένως δεν θα ανταποκρινόταν στις πραγματικές ανάγκες, επιλέχθηκε η δημιουργία ενός MVP το οποίο θα λειτουργούσε ως μηχανισμός δοκιμής υποθέσεων, όπως ακριβώς προτείνει η *Lean Startup* προσέγγιση.

Η εφαρμογή του κύκλου *Build–Measure–Learn* επιβεβαιώθηκε σε όλα τα στάδια: πρώτα υλοποιήθηκε ο βασικός πυρήνας και στην συνέχεια δημιουργήθηκαν μαθήματα για την επέκταση της πλατφόρμας. Με αυτόν τον τρόπο, η μεθοδολογία MVP δεν εφαρμόστηκε ως θεωρητική αναφορά, αλλά ως βασική στρατηγική λήψης αποφάσεων και καθοδήγησης της ανάπτυξης.

Στο πλαίσιο της παρούσας εργασίας, το MVP σχεδιάστηκε με σκοπό να καλύψει τις βασικές ανάγκες τριών βασικών ρόλων: μαθητών, εκπαιδευτών και διαχειριστή (admin). Το ελάχιστο βιώσιμο προϊόν περιλαμβάνει τις κρίσιμες λειτουργίες που πρέπει να υποστηρίζει η πλατφόρμα, ώστε να μπορεί να λειτουργήσει αποδοτικά σε περιβάλλον δοκιμών ή παραγωγής. Αναλυτικότερα για την ανάπτυξη της δικιάς μας πλατφόρμας προκύπτει ο παρακάτω πίνακας με τις απαραίτητες λειτουργίες:

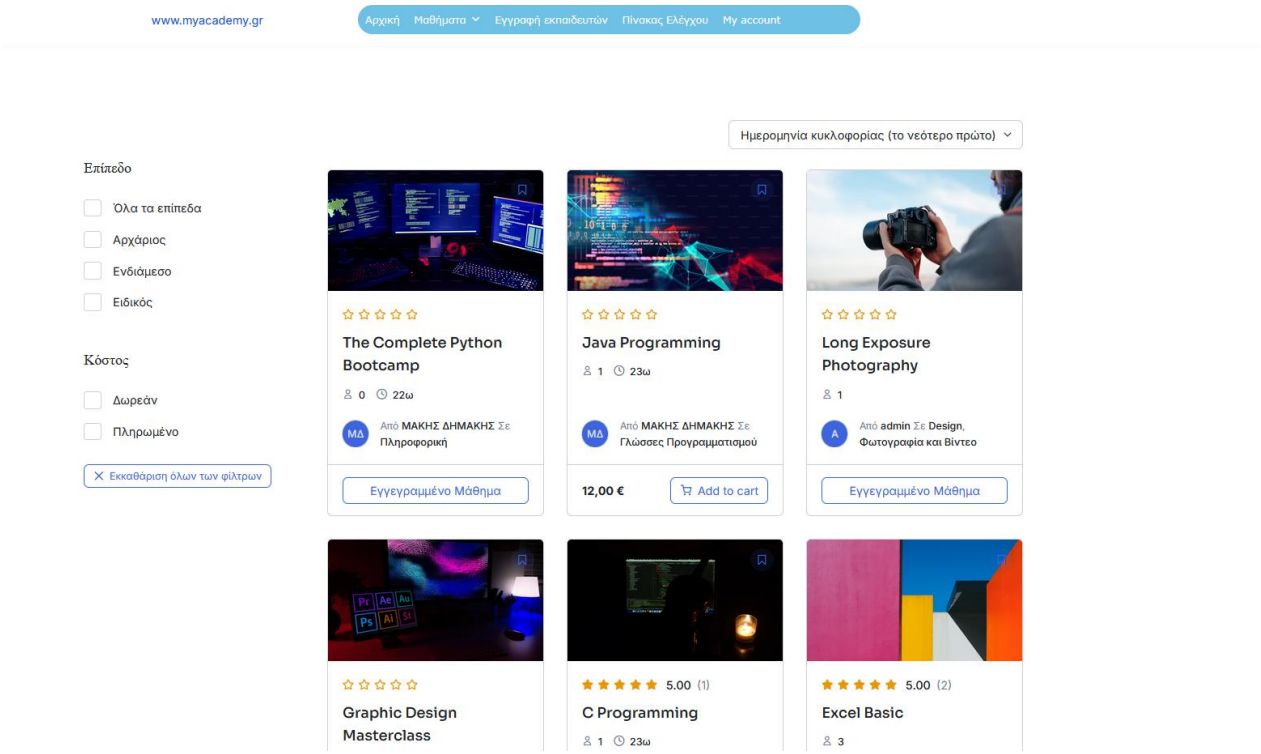
A/A	Κατηγορία	Χαρακτηριστικό
1	**Μαθητές**	Δημιουργία λογαριασμού
2		Επεξεργασία προφίλ (φωτογραφία, όνομα, στοιχεία κλπ)
3		Κατηγορίες Μαθημάτων
4		Αναζήτηση και Φίλτρα
5		Προβολή πληροφοριών για τα μαθήματα και τους εκπαιδευτές
6		Προβολή ενός εισαγωγικού βίντεο για το κάθε μάθημα
7		Προβολή αξιολογήσεων μαθημάτων
8		Εγγραφή σε μάθημα
9		Κουίζ
10		Κριτικές και Αξιολογήσεις Μαθημάτων
11		Απεγγραφή από μάθημα
12		Virtual Wallet
13	**Εκπαιδευτές**	Πίνακας Εργαλείων Εκπαιδευτών(Δημιουργία/Διαγραφή Μαθήματος κ.α.)
14		Δημιουργία λογαριασμού
15		Επεξεργασία προφίλ (φωτογραφία, όνομα, στοιχεία κλπ)
16	**Πλατφόρμα**	Πίνακας Εργαλείων διαχειρηστή
17		Έγκριση μαθημάτων πριν τη δημοσίευση
18		Έλεγχος των εκπαιδευτών (π.χ. επαλήθευση ταυτότητας)
19		Παρακολούθηση εσόδων - εξόδων
20	**Πληρωμές**	Εφάπαξ Πληρωμές
21		Λήψη πιστοποιητικών ολοκλήρωσης

Πίνακας 2.1: Βασικές λειτουργίες για την υλοποίηση του MVP

Σύνοψη & Ανάλυση των Στιγμιοτύπων της Πιλοτικής Εφαρμογής "myacademy".
Η πιλοτική εφαρμογή "myacademy" παρουσιάζει δύο διαφορετικές αρχιτεκτονικές υλοποίησης για την

ίδια πλατφόρμα e-learning:
Στιγμιότυπα από την Αρχική Σελίδα Μαθημάτων:

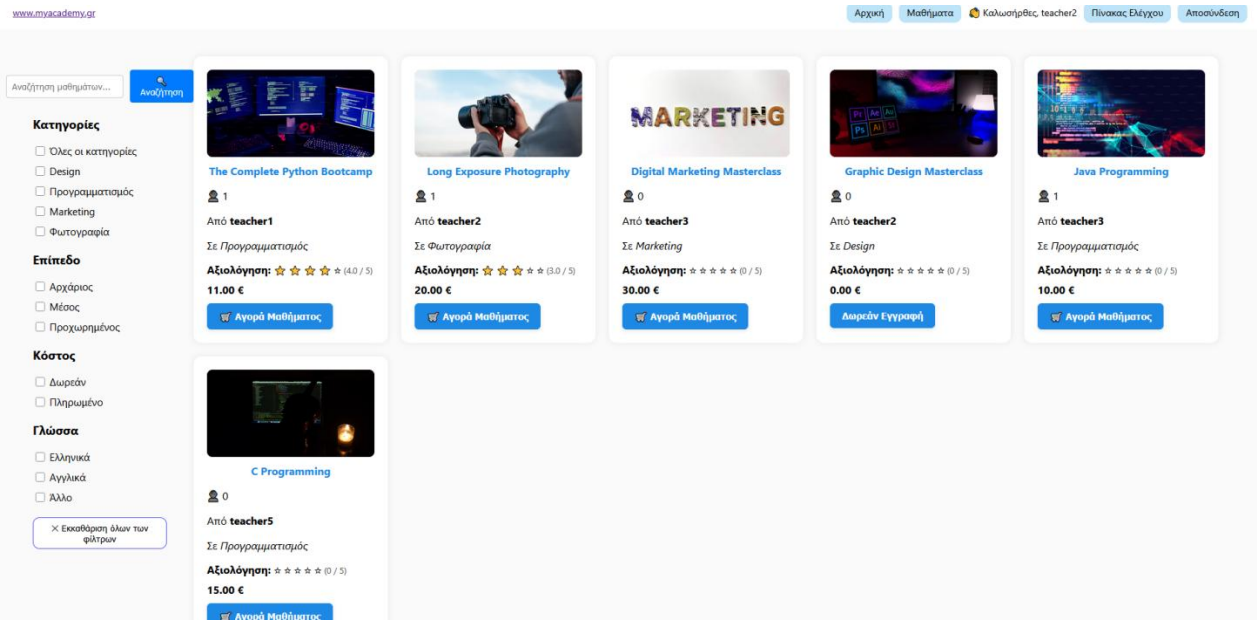
WordPress



Εικόνα 2.4: Αρχική σελίδα μαθημάτων WordPress edition

- **Δομή**:
  - **Κατηγοριοποίηση μαθημάτων** με φίλτρα (επίπεδο, κόστος).
  - **Παράδειγμα μαθήματος**:
    - *"Java Programming"* (23 ώρες, 12.00 €).
  - **Λειτουργίες**:
    - Κουμπιά "Εγγραφή" και "Add to Cart".
    - Αξιολογήσεις (π.χ. 5 ★ για το μάθημα C Programming).

Microservices



Εικόνα 2.5: Αρχική σελίδα μαθημάτων Microservices edition

- **Δομή**:
  - Πιο **δυναμική διεπαφή** με checkboxes για φιλτράρισμα (κατηγορίες, επίπεδο, κόστος, γλώσσα).
  - **Παράδειγμα μαθήματος**:
    - *"C Programming"* (30.00 €, χωρίς αξιολογήσεις).
    - *"Long Exposure Photography"* (20.00 €).
  - **Λειτουργίες**:
    - Κουμπιά "Αγορά Μαθήματος", "Δωρεάν Εγγραφή".

2.3. Περιγραφή ρόλων: Admin, Εκπαιδευτής, Μαθητής

Η αρχιτεκτονική της πλατφόρμας βασίζεται σε τρεις κύριους τύπους χρηστών, οι οποίοι αλληλεπιδρούν μεταξύ τους. Οι ρόλοι αυτοί είναι ο μαθητής (student), ο εκπαιδευτής (teacher) και ο διαχειριστής (admin). Κάθε ρόλος διαθέτει διαφορετικό επίπεδο πρόσβασης και διαφορετικά δικαιώματα, ανάλογα με τις ανάγκες που καλείται να εξυπηρετήσει.

Ο **μαθητής** είναι ο τελικός χρήστης της πλατφόρμας, ο οποίος παρακολουθεί μαθήματα και έχει πρόσβαση σε λειτουργίες αξιολόγησης. Ο **εκπαιδευτής** έχει τη δυνατότητα δημιουργίας και διαχείρισης εκπαιδευτικού περιεχομένου, ενώ μέσω ενός ειδικού πίνακα εργαλείων μπορεί να παρακολουθεί τη συμμετοχή των μαθητών του και να επεξεργάζεται πληροφορίες για τα μαθήματά του. Τέλος, ο **διαχειριστής** είναι υπεύθυνος για την εποπτεία της πλατφόρμας: διαχειρίζεται χρήστες, εγκρίνει περιεχόμενο πριν δημοσιευθεί, παρακολουθεί τα οικονομικά στοιχεία και φροντίζει για την ορθή λειτουργία του συστήματος συνολικά.

2.4. Γενικό λογικό μοντέλο λειτουργίας

Το **λογικό μοντέλο λειτουργίας** της πλατφόρμας απεικονίζεται μέσω της μεθοδολογίας **Data Flow Diagrams (DFD).**

Τα Data Flow Diagrams (DFD) αποτελούν βασικό εργαλείο του δομημένου σχεδιασμού συστημάτων (Structured Analysis) και χρησιμοποιούνται για την απεικόνιση της ροής δεδομένων μέσα σε ένα πληροφοριακό σύστημα. Όπως περιγράφει ο Tom DeMarco [9], το DFD είναι μια γραφική απεικόνιση των διαδικασιών ενός συστήματος και των δεδομένων που ανταλλάσσονται μεταξύ αυτών, των αποθηκευτικών μονάδων και των εξωτερικών οντοτήτων.

Η χρήση των DFD στοχεύει στην ανάλυση του συστήματος όχι από την πλευρά των χρηστών ή των μηχανισμών ελέγχου, αλλά από τη σκοπιά της ροής των δεδομένων. Αυτή η μεθοδολογία επιτρέπει την αναγνώριση των κρίσιμων σημείων επεξεργασίας πληροφορίας χωρίς να δεσμεύεται η ανάλυση από τεχνικά ή λειτουργικά σημεία. Ο αναλυτής προσεγγίζει το σύστημα ακολουθώντας την πορεία τους σε όλη τη διάρκεια επεξεργασίας τους μέσα στο σύστημα.

Σύμφωνα με τον DeMarco, τα DFD είναι:

- **Γραφικά**: χρησιμοποιούν σύμβολα για να αναπαραστήσουν διεργασίες, ροές, αποθηκεύσεις και εξωτερικές οντότητες.
- **Κατακερματισμένα**: υποστηρίζουν ιεραρχική ανάλυση με επίπεδα (context diagram, level 1, level 2 κ.λπ.).
- **Πολυδιάστατα**: μπορούν να παρουσιάζουν τόσο φυσικές όσο και λογικές όψεις του συστήματος.
- **Εστιασμένα στη ροή δεδομένων**: δίνουν έμφαση στην κίνηση και μετασχηματισμό των δεδομένων.
- **Μη εστιασμένα στον έλεγχο**: περιορίζουν την απεικόνιση μηχανισμών ροής ελέγχου (π.χ. βρόχους, δομές απόφασης).

Ένα τυπικό DFD περιλαμβάνει τέσσερα βασικά συστατικά:

1. **Διεργασίες (Processes)** – τι συμβαίνει στα δεδομένα.
2. **Ροές δεδομένων (Data Flows)** – πώς κινούνται τα δεδομένα.
3. **Αποθηκεύσεις (Data Stores)** – πού διατηρούνται τα δεδομένα.
4. **Εξωτερικές οντότητες (External Entities)** – ποιοι είναι οι εξωτερικοί φορείς που αλληλεπιδρούν με το σύστημα.

Η δομή και η διαδοχική ανάλυση που προσφέρουν τα DFD επιτρέπουν στον αναλυτή να δημιουργήσει μια πλήρη εικόνα του συστήματος προτού μεταβεί στην υλοποίηση. Το DFD δεν προσπαθεί να απεικονίσει τον τρόπο με τον οποίο πραγματοποιείται ο έλεγχος, αλλά δίνει σαφή έμφαση στη λειτουργική ροή των δεδομένων, αποτελώντας ένα από τα πιο ισχυρά εργαλεία του δομημένου σχεδιασμού.[10]

Τα context diagrams παρουσιάζουν τη συνολική εικόνα της πλατφόρμας ως ένα ενιαίο σύστημα (Course Academy), που δέχεται εισόδους και παρέχει εξόδους στους βασικούς τύπους χρηστών: μαθητές, εκπαιδευτές και διαχειριστές.

Στα παρακάτω διαγράμματα επιπέδου Context, απεικονίζεται η βασική λειτουργική σχέση μεταξύ των βασικών ρόλων της πλατφόρμας (εξωτερικές οντότητες – actors) και του συστήματος (**Course Academy**). Το σύστημα αναπαριστάται ως μία οντότητα «μαύρου κουτιού», ενώ οι αλληλεπιδράσεις με τους χρήστες περιγράφονται μέσω ροών δεδομένων.

Οι κύριοι ρόλοι είναι:

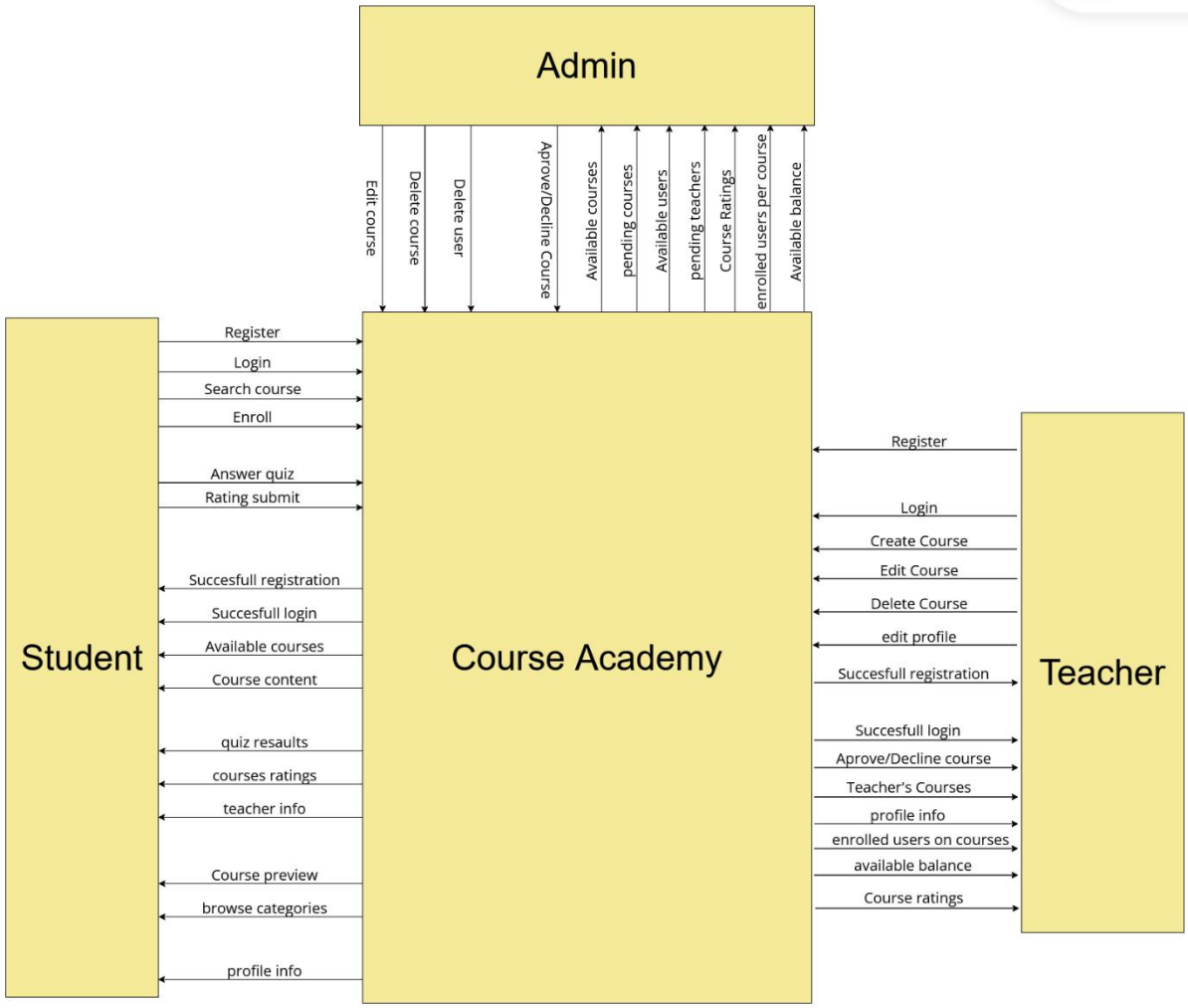
- **Student (Μαθητής):** Ο τελικός χρήστης που εγγράφεται στην πλατφόρμα, παρακολουθεί μαθήματα, απαντά σε κουίζ και λαμβάνει πιστοποιήσεις.
- **Teacher (Εκπαιδευτής):** Δημιουργεί και διαχειρίζεται μαθήματα, βλέπει στατιστικά και μπορεί να επεξεργάζεται προφίλ και περιεχόμενο.
- **Admin (Διαχειριστής):** Εποπτεύει συνολικά το σύστημα, εγκρίνει χρήστες ή μαθήματα, έχει δικαιώματα διαγραφής και μπορεί να δει στατιστικά ή οικονομικά στοιχεία.

Τα **βελάκια** αναπαριστούν τις ροές δεδομένων (data flows) μεταξύ των χρηστών και του συστήματος, όπως αιτήματα (requests), πληροφορίες, υποβολές ή απαντήσεις.

Για παράδειγμα:

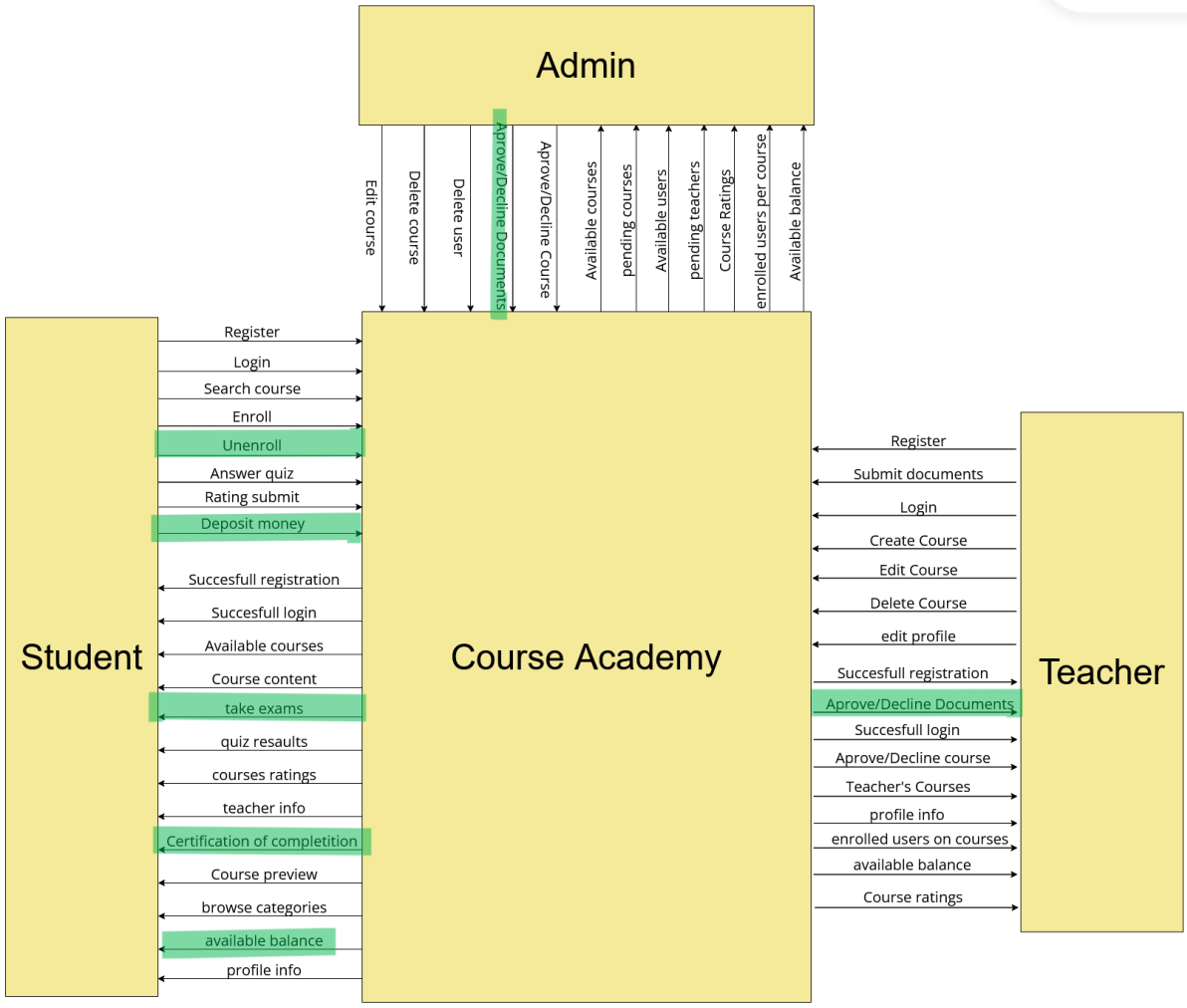
- **Βέλος από Student → System**:
  - *"Εγγραφή σε μάθημα"* (ο Student στέλνει αίτημα στο σύστημα).
- **Βέλος από System → Teacher**:
  - *"Ειδοποίηση για νέα αξιολόγηση"* (το σύστημα ενημερώνει τον Teacher).
- **Βέλος από Admin → System**:
  - *"Διαγραφή μαθήματος"* (ο Admin εκτελεί ενέργεια στο σύστημα).

Wordpress Context Diagram



Εικόνα 2.6: WordPress DFD Context Diagram

Microservices Context Diagram



Εικόνα 2.7: Microservices DFD Context Diagram

Χρωματισμένες με πράσινο χρώμα είναι οι πρόσθετες λειτουργίες που θα υλοποιηθούν με την αρχιτεκτονική των microservices.

Οι βασικές ροές δεδομένων είναι οι εξής:

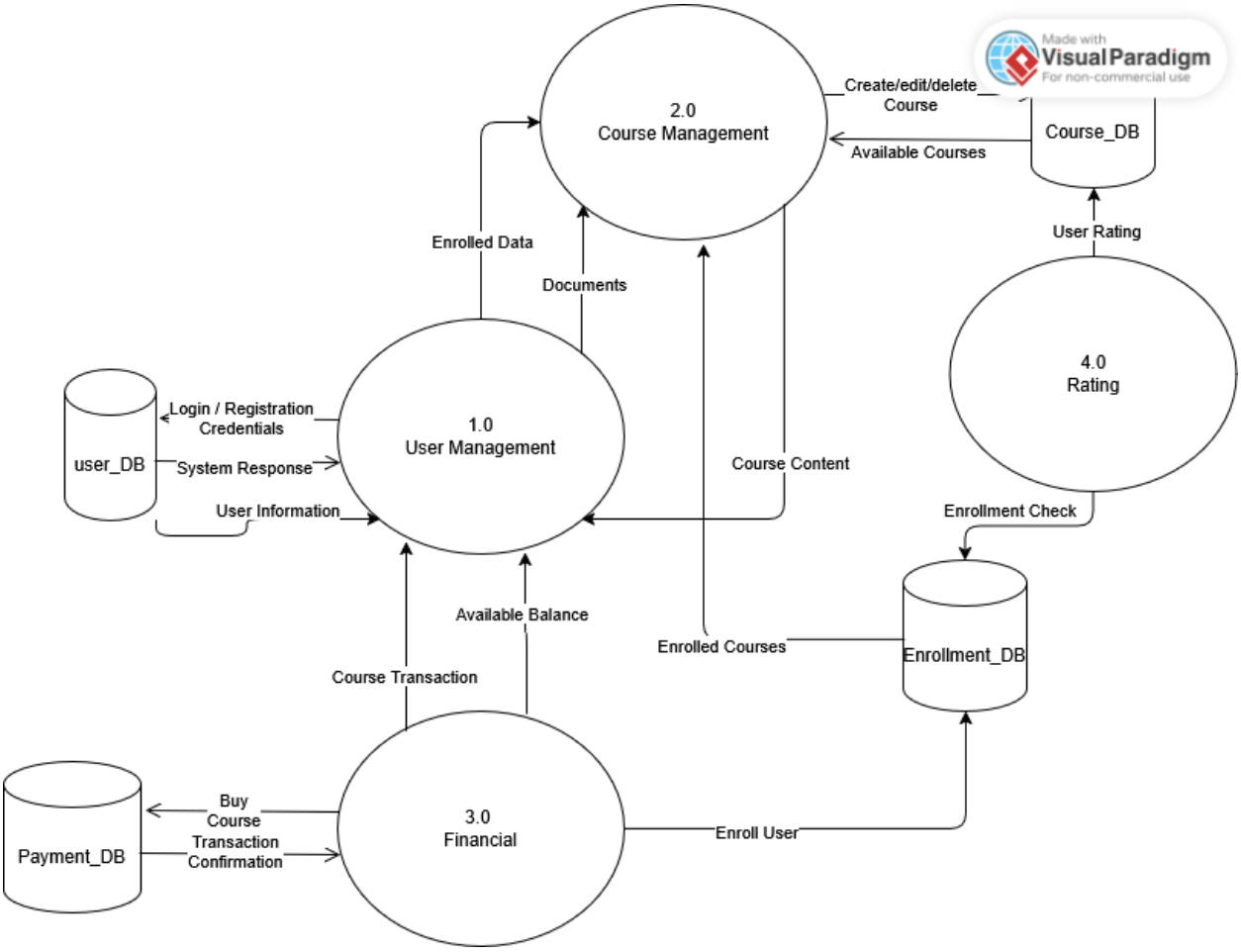
- Οι μαθητές αλληλεπιδρούν με τη λειτουργία εγγραφής, αναζήτησης μαθημάτων, συμμετοχής σε quiz, εγγραφής ή απεγγραφής από μαθήματα, αξιολόγησης μαθημάτων, καθώς και με λειτουργίες

που σχετίζονται με την οικονομική διαχείριση (π.χ. φόρτιση λογαριασμού, πιστοποιητικά ολοκλήρωσης) κ.α.

- Οι εκπαιδευτές δημιουργούν, επεξεργάζονται και διαγράφουν μαθήματα, διαχειρίζονται το προφίλ τους, παρακολουθούν εγγεγραμμένους χρήστες στα μαθήματά τους και βλέπουν τη βαθμολογία και το διαθέσιμο υπόλοιπο των μαθημάτων τους. Στο διάγραμμα της microservices αρχιτεκτονικής, επιπλέον, υποβάλλουν έγγραφα προς έγκριση.
- Ο διαχειριστής (Admin) είναι υπεύθυνος για την έγκριση μαθημάτων και εκπαιδευτών, τη διαγραφή χρηστών ή μαθημάτων, καθώς και για την παρακολούθηση στατιστικών, κίνησης μαθημάτων και οικονομικών δεδομένων.

Τα context diagrams λειτουργούν ως πλαίσιο υψηλού επιπέδου για το πώς διασυνδέονται όλοι οι χρήστες με τις βασικές διεργασίες του συστήματος.

Παρουσιάζεται το **DFD επιπέδου 0** με τις κυριες λειτουργίες και την αλληλεπίδραση μεταξύ τους:



Εικόνα 2.8: DFD Διάγραμμα επιπέδου 0

Στη συνέχεια, τα **DFD επιπέδου 1**, τα οποία περιλαμβάνονται στα κεφάλαια 4.2.1 και 4.3.1, εξειδικεύουν τη ροή δεδομένων ανά τεχνολογική υλοποίηση (WordPress και Microservices αντίστοιχα), παρουσιάζοντας τις εσωτερικές διεργασίες και βάσεις δεδομένων που συμμετέχουν στην κάθε λειτουργία.

Η χρήση των DFD εξασφαλίζει τη σαφή κατανόηση του τρόπου με τον οποίο οι χρήστες αλληλεπιδρούν με την πλατφόρμα, λειτουργεί ως οδηγός για την υλοποίηση και αποτελεί απαραίτητο εργαλείο στη διαδικασία σχεδιασμού σύμφωνα με τη μεθοδολογία **System Analysis & Design**.

3. Τεχνολογική Θεμελίωση

3.1. Επισκόπηση WordPress ως CMS

Το **WordPress** [11] είναι ένα από τα πιο διαδεδομένα **Συστήματα Διαχείρισης Περιεχομένου (Content Management Systems – CMS)** παγκοσμίως, γνωστό για την ευχρηστία του, την ευελιξία παραμετροποίησης και την τεράστια κοινότητα υποστήριξης. Αρχικά σχεδιάστηκε για blogging, ωστόσο έχει εξελιχθεί σε μια πλήρη πλατφόρμα για την κατασκευή ιστοσελίδων κάθε είδους, συμπεριλαμβανομένων ηλεκτρονικών καταστημάτων, ειδησεογραφικών portals και όπως στην παρούσα εργασία, για δημιουργία εκπαιδευτικών πλατφορμών.

Η βασική δομή του WordPress βασίζεται σε μια **ιεραρχία σελίδων(pages) και αναρτήσεων (posts)**, τις οποίες ο διαχειριστής μπορεί να δημιουργεί και να οργανώνει μέσω ενός φιλικού περιβάλλοντος διαχείρισης. Τα δεδομένα αποθηκεύονται σε **βάση δεδομένων MySQL**, ενώ η λειτουργικότητα και η εμφάνιση του ιστότοπου διαμορφώνονται μέσω **θεμάτων (themes)** και **πρόσθετων (plugins)**.

Ένα από τα βασικά πλεονεκτήματα του WordPress είναι η **επεκτασιμότητα μέσω plugins**, τα οποία επιτρέπουν την ενσωμάτωση επιπλέον λειτουργιών χωρίς ανάγκη ανάπτυξης από το μηδέν. Στην περίπτωση μιας πλατφόρμας διαδικτυακής εκπαίδευσης, αυτό σημαίνει ότι μπορεί να προστεθεί υποστήριξη για:

- διαχείριση μαθημάτων,
- εγγραφές χρηστών με ρόλους (π.χ. μαθητής, καθηγητής),
- ηλεκτρονικές πληρωμές,
- quizzes,
- παρακολούθηση προόδου,
- και χορήγηση πιστοποιητικών.

Η επιλογή του WordPress για μία από τις δύο υλοποιήσεις της παρούσας εργασίας έγινε με κριτήριο τη **χαμηλή τεχνική απαίτηση για ανάπτυξη**, τον γρήγορο χρόνο παράδοσης, και την ύπαρξη έτοιμων εργαλείων που επιτρέπουν την υλοποίηση ενός MVP με περιορισμένους πόρους. Στα επόμενα υποκεφάλαια αναλύονται οι βασικές τεχνολογικές αρχές λειτουργίας των plugins και παρουσιάζονται εκείνα που χρησιμοποιήθηκαν στο πλαίσιο αυτής της υλοποίησης.

3.1.1. Έννοια και ρόλος των plugins

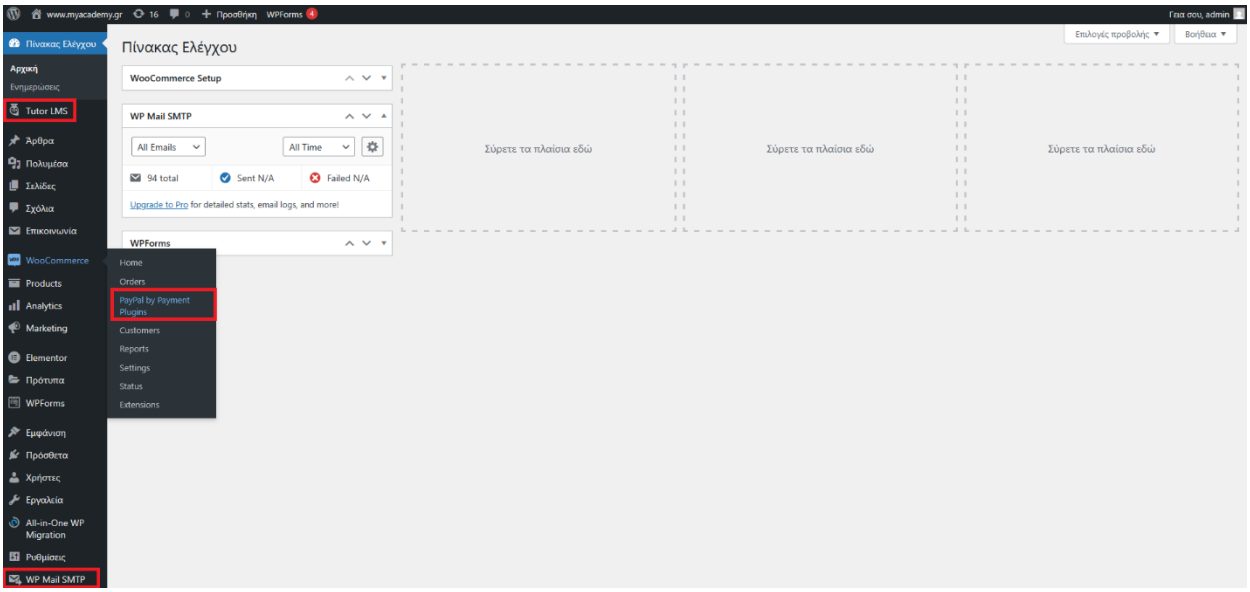
Τα plugins στο WordPress αποτελούν επεκτάσεις του βασικού του πυρήνα και χρησιμοποιούνται για την προσθήκη νέων λειτουργιών ή την προσαρμογή της συμπεριφοράς του συστήματος, χωρίς να απαιτείται επεξεργασία του ίδιου του κώδικα. Λειτουργούν ως ανεξάρτητες μονάδες λογισμικού που ενσωματώνονται δυναμικά στην πλατφόρμα, επιτρέποντας στον διαχειριστή να εμπλουτίζει την ιστοσελίδα με νέα χαρακτηριστικά ανάλογα με τις ανάγκες του. Η ύπαρξη χιλιάδων διαθέσιμων plugins

επιτρέπει στο WordPress να λειτουργήσει όχι μόνο ως σύστημα διαχείρισης περιεχομένου για απλές ιστοσελίδες, αλλά και ως ολοκληρωμένο εργαλείο για πιο σύνθετες εφαρμογές. Μέσα από τα plugins, μπορούν να ενσωματωθούν μηχανισμοί για διαχείριση μαθημάτων, αξιολόγηση μαθητών, έκδοση πιστοποιητικών, ρυθμίσεις πληρωμών, διαχείριση χρηστών με διαφορετικούς ρόλους, καθώς και υποστήριξη για κουίζ, αποστολή ειδοποιήσεων ή αναλυτικά στατιστικά χρήσης. Η δυνατότητα εγκατάστασης και ενεργοποίησης τέτοιων λειτουργιών με ελάχιστο τεχνικό κόστος καθιστά τα plugins ως σημαντικό παράγοντα της ευελιξίας του WordPress, ειδικά όταν πρόκειται για την υλοποίηση εφαρμογών με συγκεκριμένες απαιτήσεις, όπως είναι μια εκπαιδευτική πλατφόρμα.

3.1.2. Επιλογή και ανάλυση των plugins που χρησιμοποιήθηκαν

Για την υλοποίηση της πλατφόρμας διαδικτυακών μαθημάτων με χρήση WordPress, επιλέχθηκε το plugin **Tutor LMS**, το οποίο αποτέλεσε τη βασική τεχνολογική λύση για τη διαχείριση εκπαιδευτικού περιεχομένου και ροής μάθησης. Το Tutor LMS είναι ένα πλήρες σύστημα διαχείρισης μάθησης (Learning Management System) που επιτρέπει τη δημιουργία και οργάνωση μαθημάτων, την προσθήκη ενοτήτων και κουίζ, την παρακολούθηση της προόδου των μαθητών. Παράλληλα, προσφέρει λειτουργίες διαχείρισης χρηστών με ρόλους (μαθητής, εκπαιδευτής, διαχειριστής), δυνατότητα σχολίων και αξιολογήσεων, και ένα φιλικό περιβάλλον για εκπαιδευτές χωρίς απαιτήσεις προγραμματισμού.

Συμπληρωματικά, χρησιμοποιήθηκαν plugins για τη διαχείριση πληρωμών (PayPal), ώστε να υποστηρίζονται οικονομικές συναλλαγές. Επίσης, ενσωματώθηκε μηχανισμός αποστολής ειδοποιήσεων (email) για βασικές ενέργειες, όπως εγγραφή. Η επιλογή των συγκεκριμένων plugins έγινε με βάση τη σταθερότητα, τη συμβατότητα και την τεκμηρίωσή τους, ενώ ο συνδυασμός τους επέτρεψε τη γρήγορη υλοποίηση ενός λειτουργικού MVP με υψηλή επεκτασιμότητα.



Εικόνα 3.1: Πίνακας ελέγχου διαχειριστή στην έκδοση WordPress

3.2. Επισκόπηση αρχιτεκτονικής Microservices

Η αρχιτεκτονική των **Microservices** [12] αποτελεί μία από τις σημαντικότερες εξελίξεις στον χώρο του λογισμικού, προσφέροντας μια εναλλακτική προσέγγιση στη σχεδίαση πολύπλοκων εφαρμογών, σε αντίθεση με τις παραδοσιακές **μονολιθικές αρχιτεκτονικές**. Σε ένα μονολιθικό σύστημα, όλη η λειτουργικότητα υλοποιείται μέσα σε έναν ενιαίο κώδικα, όπου τα υποσυστήματα (όπως οι χρήστες, τα μαθήματα, τα κουίζ, οι πληρωμές) αλληλεξαρτώνται μεταξύ τους. Αυτό έχει ως αποτέλεσμα, κάθε αλλαγή ή αναβάθμιση να επηρεάζει ολόκληρο το σύστημα, αυξάνοντας σημαντικά τον κίνδυνο σφαλμάτων και μειώνοντας την ευελιξία.

Αντίθετα, στην αρχιτεκτονική Microservices, η εφαρμογή διασπάται σε **μια σειρά από αυτόνομες υπηρεσίες (services)**, καθεμία εκ των οποίων είναι υπεύθυνη για έναν πολύ συγκεκριμένο τομέα λειτουργίας. Κάθε μικροϋπηρεσία υλοποιείται, αναπτύσσεται, αναβαθμίζεται και συντηρείται **ανεξάρτητα**, ενώ επικοινωνεί με τις υπόλοιπες μέσω **HTTP REST APIs**. Επομένως, η διαχείριση χρηστών, η εγγραφή σε μαθήματα, τα κουίζ και οι πληρωμές μπορούν να υλοποιούνται ως ξεχωριστές υπηρεσίες, με το δικό τους σύστημα αποθήκευσης (βάση δεδομένων), λογική και αυτονομία.

Η προσέγγιση αυτή προσφέρει πληθώρα πλεονεκτημάτων. Καταρχάς, επιτρέπει σε μεγάλες ομάδες ανάπτυξης να εργάζονται ταυτόχρονα σε διαφορετικά μέρη του συστήματος, χωρίς να επηρεάζει η μία την άλλη. Επιπλέον, καθιστά ευκολότερη τη κλιμάκωση επιμέρους υπηρεσιών, ανάλογα με τις ανάγκες απόδοσης. Για παράδειγμα, αν η υπηρεσία quiz χρησιμοποιείται εντατικά, μπορεί να αναπτυχθεί χωρίς να επηρεαστεί η υπηρεσία πληρωμών. Επίσης, προσφέρει τη δυνατότητα το κάθε microservice να μπορεί να αναπτυχθεί με διαφορετική γλώσσα προγραμματισμού, framework ή βάση δεδομένων, ανάλογα με τις απαιτήσεις.

Ωστόσο, τα microservices έχουν σημαντικές προκλήσεις. Η πολυπλοκότητα μεταφέρεται από τον κώδικα σε επίπεδο αρχιτεκτονικής. Προκύπτουν ανάγκες για μηχανισμούς συγχρονισμού, παρακολούθησης, διαχείρισης σφαλμάτων και ασφαλούς επικοινωνίας μεταξύ υπηρεσιών. Απαιτείται επίσης πιο σύνθετη στρατηγική deployment (π.χ. Docker, Kubernetes), καθώς και πλήρης αυτοματοποίηση διαδικασιών (CI/CD) για να διασφαλιστεί η σταθερότητα σε ένα δυναμικά εξελισσόμενο περιβάλλον.

Στο πλαίσιο της παρούσας εργασίας, η προσέγγιση των microservices υιοθετείται ως εναλλακτική της υλοποίησης με WordPress, με σκοπό τη δημιουργία μιας αρχιτεκτονικής που ανταποκρίνεται στις απαιτήσεις επεκτασιμότητας, ανεξαρτησίας και λειτουργικής αυτονομίας. Η πλατφόρμα αναλύθηκε και διασπάστηκε σε βασικές λειτουργικές μονάδες: κεντρική υπηρεσία διαχείρισης αιτημάτων, υπηρεσία χρηστών, υπηρεσία μαθημάτων, υπηρεσία εγγραφών και υπηρεσία πληρωμών, καθεμία από τις οποίες αποτελεί ξεχωριστό microservice. Στα επόμενα υποκεφάλαια αναλύεται η μεταξύ τους επικοινωνία, οι τεχνικές λύσεις συγχρονισμού και οι μηχανισμοί εξουσιοδότησης που εφαρμόστηκαν για την ασφάλεια και αξιοπιστία της συνολικής αρχιτεκτονικής.

3.2.1. Επικοινωνία και συγχρονισμός μεταξύ υπηρεσιών

Στην παρούσα αρχιτεκτονική microservices η γλώσσα προγραμματισμού που επιλέχθηκε είναι η Python και χρησημοποιήθηκε το Django framework, η επικοινωνία μεταξύ των επιμέρους υπηρεσιών πραγματοποιείται μέσω **RESTful APIs**, με χρήση HTTP πρωτοκόλλου και ανταλλαγή δεδομένων σε μορφή **JSON**. Η κάθε μικροϋπηρεσία (main, users, courses, enrollment, payment) λειτουργεί ως ανεξάρτητη οντότητα, με σαφώς ορισμένα endpoints και τη δική της βάση δεδομένων. Η ευθύνη για τη συντονισμένη συνεργασία τους αναλαμβάνεται από την κεντρική υπηρεσία **myacademy(main)**, η οποία είναι το **gateway** και χρησιμοποιεί τη λογική αποστολής αιτημάτων και δεδομένων στα κατάλληλα microservices μέσω **proxy views**.

Οι **proxy** μηχανισμοί που υλοποιούνται στο views.py της εφαρμογής myacademy αναλαμβάνουν να μεταβιβάζουν αιτήματα από το frontend προς τις κατάλληλες υπηρεσίες, χειριζόμενοι την εξουσιοδότηση μέσω tokens και την ανακατεύθυνση αιτήσεων. Για παράδειγμα, η λειτουργία proxy_courses προωθεί αιτήματα προς την υπηρεσία διαχείρισης μαθημάτων (courses) διατηρώντας τις επικεφαλίδες (Bearer) το Token αυθεντικοποίησης, ενώ η proxy_users εξυπηρετεί αιτήματα σχετικά με σύνδεση, εγγραφή ή λήψη στοιχείων χρηστών. Αυτός ο σχεδιασμός κρύβει τη σύνθετη αρχιτεκτονική από τον τελικό χρήστη, ο οποίος αλληλεπιδρά μόνο με το myacademy, σαν να πρόκειται για ενιαίο σύστημα.

Η μεταξύ των υπηρεσιών επικοινωνία **παραμένει συγχρονισμένη**, βασισμένη σε μοντέλο αιτήματος–απάντησης, χωρίς χρήση μηνυμάτων ή queues. Τα επιμέρους services ανταλλάσσουν δεδομένα μόνο όταν είναι απαραίτητο, με χρήση αναγνωριστικών (όπως user_id, course_id), ώστε να διατηρείται ο χαμηλός βαθμός συσχέτισης. Όλες οι ροές όπως εγγραφή σε μάθημα μετά από πληρωμή ή επιβεβαίωση ταυτότητας χρήστη γίνονται κεντρικά από το myacademy, το οποίο εκτελεί τις κατάλληλες κλήσεις με τη σωστή σειρά.

Η ύπαρξη των proxies **προσφέρει διαχωρισμό μεταξύ παρουσίασης και λογικής,** διατηρώντας την αυτονομία των services και επιτρέποντας στο frontend να λειτουργεί χωρίς να γνωρίζει τον εσωτερικό καταμερισμό λειτουργιών. Το αποτέλεσμα είναι ένα κατανεμημένο αλλά συνεκτικό σύστημα, στο οποίο οι υπηρεσίες συνεργάζονται αποτελεσματικά, διατηρώντας παράλληλα ανεξαρτησία ανάπτυξης και συντήρησης.

3.2.2. Διαχείριση δεδομένων και ενορχήστρωση

Στο σύστημα που αναπτύχθηκε, η κάθε μικροϋπηρεσία διαχειρίζεται **αποκλειστικά τα δικά της δεδομένα**. Η απομόνωση αυτή ενισχύει την ανεξαρτησία των υπηρεσιών, την ασφάλεια των δεδομένων και τη δυνατότητα αυτόνομης ανάπτυξης και συντήρησης.

Η υπηρεσία **users** είναι υπεύθυνη για την αποθήκευση και διαχείριση των στοιχείων των χρηστών. Περιλαμβάνει λειτουργίες όπως εγγραφή, σύνδεση, προβολή προφίλ και διαχείριση ρόλων. Όλα τα δεδομένα των χρηστών (credentials, ρόλοι, tokens κ.λπ.) αποθηκεύονται στη δική της βάση, και οι υπόλοιπες υπηρεσίες δεν έχουν άμεση πρόσβαση σε αυτή.

Η υπηρεσία **courses** διαχειρίζεται τα μαθήματα. Κάθε μάθημα αποθηκεύεται μαζί με τίτλο, περιγραφή, κατηγορία και αναφορά στον δημιουργό (teacher), μέσω του user_id. Η courses δεν επικοινωνεί άμεσα με την users, αλλά λαμβάνει τα IDs μέσω αιτημάτων που διεκπεραιώνει το API Gateway (myacademy), το οποίο φέρει την ευθύνη για τον έλεγχο ταυτότητας.

Η υπηρεσία **enrollment** καταγράφει τις εγγραφές μαθητών σε μαθήματα, διατηρώντας σχέσεις τύπου user_id – course_id, ώστε να ελέγχει ποιοι χρήστες έχουν πρόσβαση σε ποια μαθήματα. Είναι δηλαδή ο ενδιάμεσος σύνδεσμος μεταξύ του περιεχομένου και του χρήστη. Παρότι δεν αντλεί άμεσα πληροφορίες από τα users ή courses, βασίζεται στα IDs που λαμβάνει από το frontend μέσω myacademy.

Η υπηρεσία **payment** καταγράφει τις οικονομικές συναλλαγές, επεξεργάζεται αιτήματα πληρωμών και ενημερώνει την enrollment για ενεργοποίηση πρόσβασης σε μαθήματα μόνο αφού ολοκληρωθεί επιτυχώς η πληρωμή. Δεν αποθηκεύει εκπαιδευτικά ή προσωπικά δεδομένα, αλλά εστιάζει σε συναλλαγές και αποτελέσματα πληρωμής.

Η **ενορχήστρωση** του συστήματος υλοποιείται **χωρίς κεντρικό orchestrator**, αλλά μέσω των **proxy views στο myacademy**, τα οποία συνθέτουν τις επιμέρους ενέργειες σε ολοκληρωμένες λειτουργικές ροές. Για παράδειγμα, σε μια διαδικασία πληρωμής και εγγραφής, το frontend επικοινωνεί μόνο με το myacademy, το οποίο εκτελεί εσωτερικά τη σειρά: αποστολή αιτήματος στην payment, λήψη αποτελέσματος και, αν η πληρωμή είναι επιτυχής, ενημέρωση της enrollment.

Ο διαχωρισμός των δεδομένων και η ανεξαρτησία των υπηρεσιών εξασφαλίζουν ότι κάθε μονάδα μπορεί να τροποποιείται ή να αναπτύσσεται χωρίς να επηρεάζεται το υπόλοιπο σύστημα.

3.2.3. Μηχανισμοί αυθεντικοποίησης

Η ασφάλεια σε μια αρχιτεκτονική microservices απαιτεί τη σαφή διάκριση μεταξύ **αυθεντικοποίησης (authentication)** και **εξουσιοδότησης (authorization)**, καθώς κάθε υπηρεσία λειτουργεί αυτόνομα και πρέπει να ελέγχει την εγκυρότητα των εισερχόμενων αιτημάτων. Στο πλαίσιο της παρούσας εφαρμογής, η αυθεντικοποίηση επιτυγχάνεται με χρήση **JSON Web Tokens (JWT)**, τα οποία εκδίδονται από την υπηρεσία users κατά τη διαδικασία σύνδεσης και μεταβιβάζονται μέσω HTTP headers σε όλα τα επόμενα αιτήματα.

Συγκεκριμένα, όταν ένας χρήστης πραγματοποιεί επιτυχώς login, η υπηρεσία users εκδίδει ένα JWT, το οποίο περιέχει πληροφορίες όπως το user_id, τον ρόλο του χρήστη (π.χ. student, teacher, admin), καθώς και χρονικό όριο ισχύος (expiration time). Το token αποθηκεύεται προσωρινά στο frontend και μεταφέρεται σε όλα τα μελλοντικά αιτήματα μέσα από το header.

Η εφαρμογή myacademy, η οποία λειτουργεί ως gateway, αναλαμβάνει τον κεντρικό έλεγχο αυθεντικοποίησης, προτού προωθήσει αιτήματα στις επιμέρους υπηρεσίες. Κάθε proxy view διαβάζει το token από το αίτημα και είτε το μεταφέρει αυτούσιο στο backend είτε το απορρίπτει αν απουσιάζει ή είναι άκυρο. Έτσι διασφαλίζεται ότι κανένα αίτημα δεν προωθείται προς τα internal APIs χωρίς επιβεβαιωμένη ταυτότητα.

Η εξουσιοδότηση επιτυγχάνεται κυρίως μέσω ελέγχου ρόλων, οι οποίοι είναι ενσωματωμένοι στο payload του JWT. Για παράδειγμα, συγκεκριμένα endpoints της υπηρεσίας courses επιτρέπεται να καλούνται μόνο από χρήστες με ρόλο teacher, ενώ διαχειριστικά endpoints ενεργοποιούνται αποκλειστικά για admin. Οι υπηρεσίες enrollment και payment χρησιμοποιούν επίσης το user_id που περιέχεται στο token για να διασταυρώνουν την ταυτότητα του αιτούντος και να διασφαλίσουν πως δεν υπάρχει παραβίαση στην πρόσβαση.

Κάθε μικροϋπηρεσία ελέγχει την εγκυρότητα του JWT κατά την είσοδο, χωρίς να χρειάζεται να επικοινωνήσει με την υπηρεσία users για επιβεβαίωση, γεγονός που μειώνει την εξάρτηση και ενισχύει την απόδοση. Η χρήση ενός **shared secret** για την υπογραφή των tokens εξασφαλίζει ότι όλες οι υπηρεσίες μπορούν να επαληθεύσουν την εγκυρότητα του token τοπικά.

Η προσέγγιση αυτή επιτρέπει την υλοποίηση ενός ασφαλούς και επεκτάσιμου μηχανισμού αυθεντικοποίησης, στον οποίο η ασφάλεια παραμένει κατανεμημένη αλλά συντονισμένη. Η χρήση του JWT διευκολύνει την ανεξαρτησία των υπηρεσιών, ελαχιστοποιεί την ανάγκη για συνεχή επαλήθευση από την users, και ενισχύει την απόκριση του συστήματος σε περιβάλλοντα με αυξημένο φόρτο.

4. Υλοποίηση των Δύο Πλατφορμών

4.1. Λογική σχεδίασης και δομή ρόλων της εφαρμογής

Η σχεδίαση της εκπαιδευτικής πλατφόρμας στηρίχθηκε σε μια σαφή και οργανωμένη **λογική ροής λειτουργιών**, η οποία παραμένει κοινή και για τις δύο υλοποιήσεις που παρουσιάζονται (με WordPress και με αρχιτεκτονική microservices). Η βασική ιδέα ήταν η δημιουργία ενός ευέλικτου και ρεαλιστικού περιβάλλοντος για διαδικτυακή εκπαίδευση, το οποίο εξυπηρετεί τρεις διακριτούς ρόλους: μαθητή, εκπαιδευτή και διαχειριστή.

Ο **μαθητής** μπορεί να περιηγείται σε μαθήματα, να εγγράφεται, να παρακολουθεί βίντεο, να απαντά σε quiz, να αξιολογεί το περιεχόμενο και να διαχειρίζεται το προφίλ του. Ο **εκπαιδευτής** δημιουργεί και δημοσιεύει μαθήματα, ανεβάζει περιεχόμενο και αξιολογεί τη συμμετοχή των μαθητών. Ο **διαχειριστής** έχει δικαιώματα εποπτείας: εγκρίνει ή απορρίπτει περιεχόμενο πριν δημοσιευθεί, επαληθεύει την ταυτότητα των εκπαιδευτών και παρακολουθεί τη ροή πληρωμών, εγγραφών και στατιστικών χρήσης.

Η συνολική λειτουργικότητα μοντελοποιείται μέσω του **Διαγράμματος Ροής Δεδομένων (DFD) επιπέδου 0**, το οποίο απεικονίζει συνοπτικά τις βασικές αλληλεπιδράσεις μεταξύ των χρηστών και του συστήματος, όπως: η εγγραφή και αυθεντικοποίηση, η αναζήτηση και παρακολούθηση μαθημάτων, οι πληρωμές και η απόδοση πιστοποιητικών, καθώς και η εποπτεία των ενεργειών από τον διαχειριστή. Το διάγραμμα αυτό αποτελεί τον πυρήνα της λογικής σχεδίασης και ορίζει την πληροφοριακή ροή που υλοποιήθηκε τεχνικά με διαφορετικά μέσα σε κάθε περίπτωση.

Η δομή των ρόλων σχεδιάστηκε με στόχο τη σαφή **κατανομή δικαιωμάτων**, ώστε κάθε χρήστης να έχει πρόσβαση μόνο στις λειτουργίες που του αναλογούν. Η προσέγγιση αυτή διασφαλίζει την ευχρηστία, την ασφάλεια και τη δυνατότητα επέκτασης του συστήματος σε πραγματικές συνθήκες χρήσης.

4.2. Υλοποίηση με WordPress

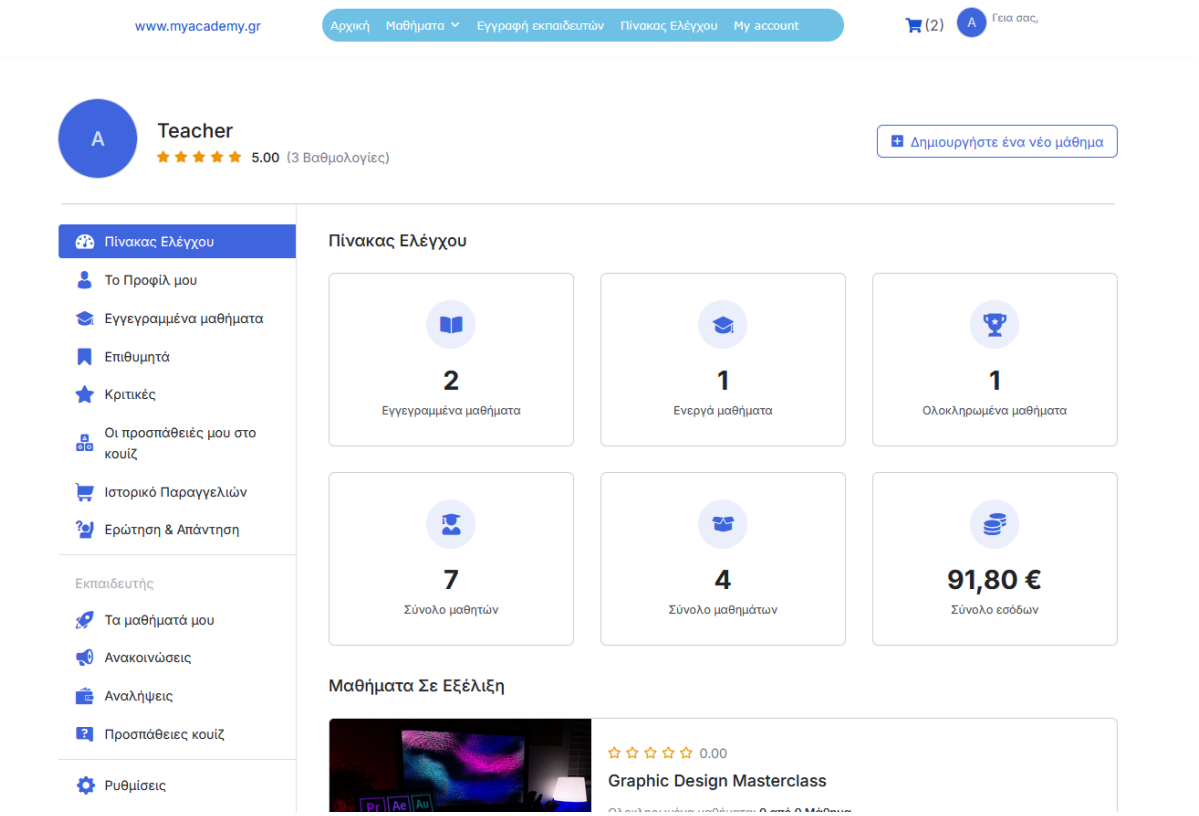
4.2.1. Αρχιτεκτονική & λειτουργική ροή

Η υλοποίηση της εκπαιδευτικής πλατφόρμας στο περιβάλλον του WordPress βασίστηκε σε μονολιθική αρχιτεκτονική, στην οποία όλες οι λειτουργίες συνυπάρχουν. Η επεκτασιμότητα και η λειτουργικότητα της πλατφόρμας επιτεύχθηκε κυρίως μέσω του πρόσθετου Tutor LMS, το οποίο παρείχε έτοιμες διεπαφές και ροές για τη διαχείριση μαθημάτων, χρηστών, αξιολογήσεων και πληρωμών.

Το σύστημα οργανώνεται γύρω από τρεις βασικούς ρόλους χρηστών: μαθητής, εκπαιδευτής, και διαχειριστής. Ο κάθε ρόλος ενεργοποιεί διαφορετικά modules του Tutor LMS. Οι μαθητές μπορούν να εγγραφούν, να περιηγηθούν στα διαθέσιμα μαθήματα, να παρακολουθήσουν περιεχόμενο, να συμμετάσχουν σε κουίζ και να αξιολογήσουν το περιεχόμενο. Οι εκπαιδευτές έχουν τη δυνατότητα δημιουργίας μαθημάτων, διαχείρισης του περιεχομένου τους και παρακολούθησης στατιστικών. Ο

διαχειριστής έχει την πλήρη εποπτεία του συστήματος, με δυνατότητα έγκρισης μαθημάτων και χρηστών, επεξεργασίας περιεχομένου και πρόσβασης σε οικονομικά δεδομένα.

Πίνακας ελέγχου χρήστη:



Εικόνα 4.1: Πίνακας ελέγχου χρήστη στην έκδοση WordPress

Η λειτουργική ροή του WordPress αποτυπώνεται στο **Διάγραμμα Ροής Δεδομένων (DFD) Επιπέδου 1**, όπου παρουσιάζονται αναλυτικά οι βασικές διεργασίες της πλατφόρμας:

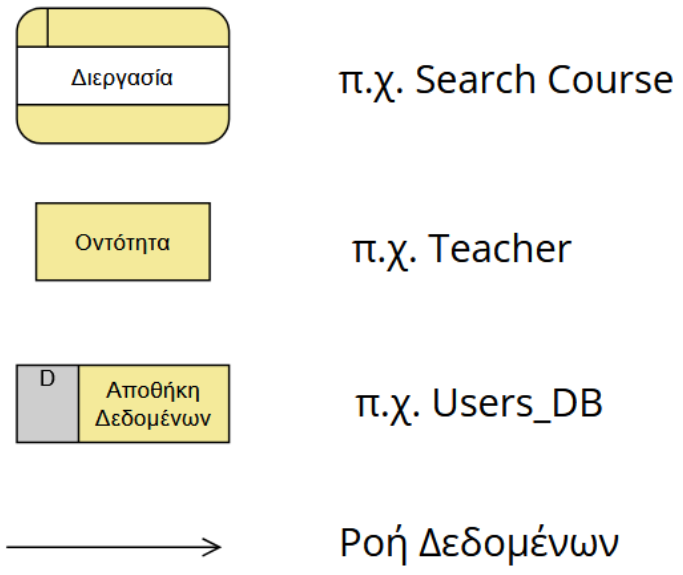
- Εγγραφή και σύνδεση χρήστη (Sign Up / Login)
- Περιήγηση και αναζήτηση μαθημάτων (Browse, Search)
- Προβολή πληροφοριών και λεπτομερειών μαθήματος
- Εγγραφή σε μάθημα, διαδικασία πληρωμής και ενεργοποίηση πρόσβασης
- Παρακολούθηση μαθήματος, συμμετοχή σε κουίζ, υποβολή απαντήσεων
- Δημιουργία, επεξεργασία και διαγραφή μαθήματος από τους εκπαιδευτές
- Εγκρίσεις και διαχειριστικές λειτουργίες από τον διαχειριστή

Οι ροές δεδομένων διασυνδέονται με πίνακες στην βάση δεδομένων του WordPress, οι οποίοι έχουν επεκταθεί από το plugin (π.χ. χρήστες, εγγραφές, βαθμολογίες, πληρωμές).

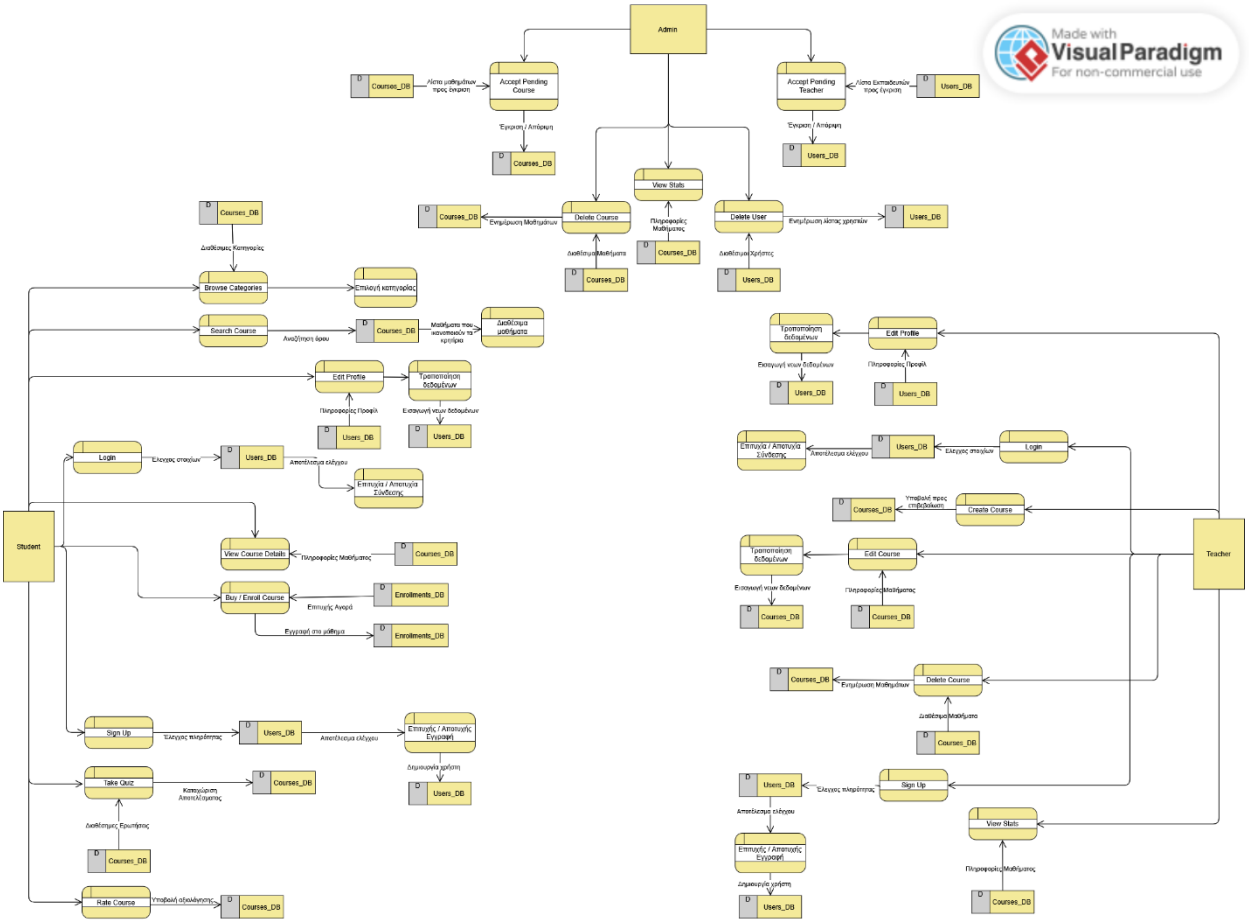
Το παρακάτω **DFD διάγραμμα**, αποτυπώνει τη ροή των βασικών λειτουργιών της πλατφόρμας. Στην αρχιτεκτονική του WordPress υλοποιήθηκαν όλες οι λειτουργίες πλήρως κατά τη φάση του MVP, όπως: εγγραφή χρηστών, αγορά μαθημάτων, συμμετοχή σε κουίζ κ.α. Άλλες λειτουργίες όπως αναφορές ή προηγμένα στατιστικά, παραμένουν προς μελλοντική ανάπτυξη.

Η επιλογή χρήσης του Tutor LMS επιτάχυνε σημαντικά τον κύκλο ανάπτυξης, αλλά ταυτόχρονα απαίτησε προσοχή στη συμβατότητα μεταξύ προσθέτων, στη διαχείριση ρυθμίσεων ασφαλείας και στη διατήρηση της συνοχής του UI/UX της πλατφόρμας.

Στο παρακάτω DFD διάγραμμα τα σχήματα που χρησιμοποιούνται είναι τα εξής:



WordPress DFD Diagram:



Εικόνα 4.2: DFD Διάγραμμα έκδοσης WordPress

4.2.2. Plugins και βασικές λειτουργικές μονάδες (modules)

Η λειτουργικότητα της πλατφόρμας στην υλοποίηση με WordPress βασίστηκε κυρίως στην αξιοποίηση του Tutor LMS, ενός ολοκληρωμένου plugin διαχείρισης εκπαιδευτικού περιεχομένου (Learning Management System), σε συνδυασμό με επιλεγμένα πρόσθετα που κάλυψαν τις ανάγκες του MVP. Μέσα από τα plugins αυτά διαμορφώθηκαν οι κύριες λειτουργίες της εφαρμογής, χωρίς να απαιτηθεί custom ανάπτυξη.

Το **Tutor LMS** είναι υπεύθυνο για τη διαχείριση της εκπαιδευτικής εμπειρίας, επιτρέποντας τη δημιουργία μαθημάτων, την οργάνωση σε ενότητες και κουίζ και την παρακολούθηση της προόδου. Μέσω του ενσωματωμένου πίνακα διαχείρισης, κάθε **εκπαιδευτής** μπορεί να δημιουργήσει, τροποποιήσει και δημοσιεύσει μαθήματα, καθώς και να βλέπει στατιστικά συμμετοχής των μαθητών του. Οι **μαθητές** μπορούν να περιηγηθούν σε διαθέσιμα μαθήματα, να εγγραφούν, να παρακολουθήσουν περιεχόμενο, να ολοκληρώσουν quiz και να υποβάλουν αξιολογήσεις. Οι **διαχειριστές** έχουν τη δυνατότητα να εγκρίνουν μαθήματα, να επιβλέπουν εκπαιδευτές και να διαχειρίζονται τις ροές εγγραφών και πληρωμών.

Η λειτουργικότητα επεκτάθηκε με τη χρήση επιπλέον plugins για:

- Ενσωμάτωση πληρωμών, με plugin που υποστηρίζει εφάπαξ αγορές μαθημάτων (PayPal)
- Αποστολή ειδοποιήσεων, μέσω email (WP Mail SMTP).

Η κάθε λειτουργική μονάδα είναι πλήρως ενσωματωμένη στο περιβάλλον διαχείρισης του WordPress, προσφέροντας έναν πίνακα εργαλείων για όλους τους τύπους χρηστών. Η παραμετροποίηση έγινε με τέτοιο τρόπο ώστε να υπάρχει καθαρός **διαχωρισμός λειτουργιών**, χωρίς επικάλυψη μεταξύ ρόλων.

Παρότι η χρήση plugins απλοποίησε σημαντικά την υλοποίηση, η αξιοπιστία του συστήματος εξαρτάται άμεσα από τη σωστή επιλογή, την ενημέρωση και τη συντήρηση των προσθέτων. Επίσης, εντοπίζονται περιορισμοί σε περιπτώσεις που απαιτείται πλήρης ελευθερία λειτουργικού σχεδιασμού, κάτι που καλύπτεται πληρέστερα στην προσέγγιση με microservices.

4.3. Υλοποίηση με Microservices

4.3.1. Περιγραφή των microservices και του ρόλου τους

Το σύστημα αποτελείται από **υπηρεσίες**, καθεμία από τις οποίες διαχειρίζεται αυτόνομα έναν κρίσιμο τομέα - σετ λειτουργιών, με δική της λογική, βάση δεδομένων και API endpoints. Οι υπηρεσίες συνεργάζονται μεταξύ τους μέσω RESTful API και καθοδηγούνται από το κεντρικό service, το myacademy, το οποίο λειτουργεί ως gateway.

Η πρώτη υπηρεσία, **users**, είναι υπεύθυνη για την αυθεντικοποίηση και διαχείριση λογαριασμών χρηστών. Περιλαμβάνει τις λειτουργίες εγγραφής, σύνδεσης (login), ενημέρωσης στοιχείων προφίλ, καθώς και αποθήκευσης του ρόλου (μαθητής ή εκπαιδευτής). Εκδίδει JWT tokens τα οποία χρησιμοποιούνται για την αυθεντικοποίηση σε όλα τα υπόλοιπα services. Η υπηρεσία ελέγχει επίσης την εγκυρότητα των credentials και εφαρμόζει τις πολιτικές ασφάλειας του συστήματος.

Η υπηρεσία **courses** διαχειρίζεται τα μαθήματα. Κάθε μάθημα συνδέεται με έναν εκπαιδευτή (μέσω user_id), περιλαμβάνει τίτλο, περιγραφή και μπορεί να τροποποιηθεί, διαγραφεί ή προβληθεί από τους μαθητές. Η υπηρεσία προσφέρει endpoints για δημιουργία, ενημέρωση και ανάκτηση μαθημάτων. Δεν διατηρεί πληροφορίες για τους εγγεγραμμένους χρήστες ούτε για τις πληρωμές.

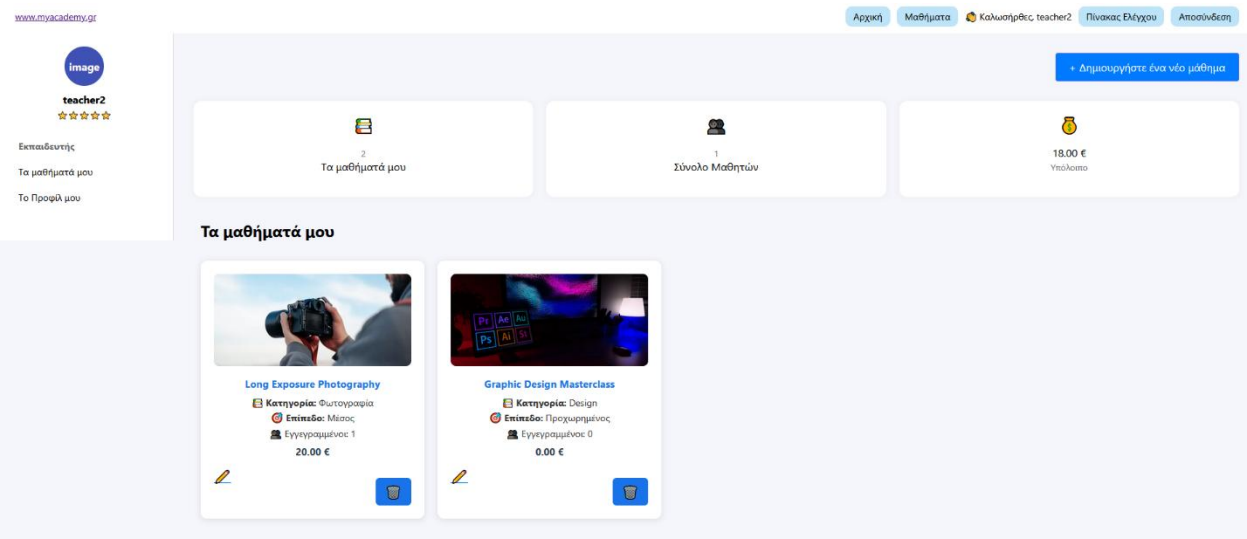
Η υπηρεσία **enrollment** λειτουργεί ως γέφυρα μεταξύ μαθητή και μαθήματος. Είναι υπεύθυνη για την αποθήκευση των εγγραφών χρηστών σε συγκεκριμένα courses και τον έλεγχο πρόσβασης. Η ενεργοποίηση μιας εγγραφής εξαρτάται από την επιβεβαίωση πληρωμής από την αντίστοιχη υπηρεσία. Η enrollment επικυρώνει κάθε φορά αν ο χρήστης είναι εγγεγραμμένος στο μάθημα, πριν επιτρέψει την πρόσβαση στο περιεχόμενο.

Η υπηρεσία **payment** χειρίζεται τις συναλλαγές. Υποστηρίζει την καταγραφή πληρωμών μαθημάτων, τη διατήρηση ενός "εικονικού πορτοφολιού" (wallet) για κάθε χρήστη, και την ενημέρωση του υπολοίπου μετά από κάθε επιτυχημένη συναλλαγή. Όταν μία πληρωμή ολοκληρώνεται, ειδοποιεί την υπηρεσία enrollment ώστε να πραγματοποιηθεί η εγγραφή του χρήστη στο μάθημα.

Τέλος, η υπηρεσία **myacademy** λειτουργεί ως ενιαίο entry point. Περιλαμβάνει proxy views που αναλαμβάνουν να δέχονται αιτήματα από το frontend, να επαληθεύουν τα tokens και να προωθούν τα αιτήματα στις αντίστοιχες μικροϋπηρεσίες. Αυτό επιτρέπει την απόκρυψη της εσωτερικής πολυπλοκότητας και προσφέρει μια ενοποιημένη διεπαφή χρήσης, ανεξαρτήτως της σύνθετης αρχιτεκτονικής.

Η επιλογή να σχεδιαστεί το σύστημα με αυτές τις πέντε κύριες υπηρεσίες βασίστηκε σε λειτουργική ανάλυση του MVP. Ο καθαρός διαχωρισμός των ευθυνών, η ανεξαρτησία βάσεων και η χρήση αυτόνομης λογικής κάθε υπηρεσίας καθιστούν την πλατφόρμα εύκολα επεκτάσιμη και συντηρήσιμη.

Πίνακας ελέγχου χρήστη:



Εικόνα 4.3: Πίνακας ελέγχου χρήστη στην έκδοση Microservices

4.3.2. Αρχιτεκτονική & λειτουργική ροή

Η λειτουργική ροή της εφαρμογής με βάση την αρχιτεκτονική microservices αποτυπώνεται στο παρακάτω **DFD διάγραμμα**, το οποίο παρουσιάζει αναλυτικά τις βασικές διεργασίες ανά ρόλο χρήστη (μαθητής, εκπαιδευτής, διαχειριστής), τις αντίστοιχες βάσεις δεδομένων και τις διασυνδέσεις μεταξύ τους.

Σε αντίθεση με τη μονολιθική λογική του WordPress, η υλοποίηση με microservices βασίζεται στη διάσπαση των λειτουργιών σε αυτόνομες υπηρεσίες. Οι χρήστες δεν αλληλεπιδρούν άμεσα με κάθε

μικροϋπηρεσία, αλλά μέσω ενός ενιαίου συστήματος (Course Academy), το οποίο αντιστοιχεί στο myacademy service της εφαρμογής. Από εκεί, τα αιτήματα δρομολογούνται στις κατάλληλες υπηρεσίες (users, courses, enrollment, payment), σύμφωνα με τον τύπο και το περιεχόμενο της ενέργειας.

Το διάγραμμα περιλαμβάνει λειτουργίες όπως:

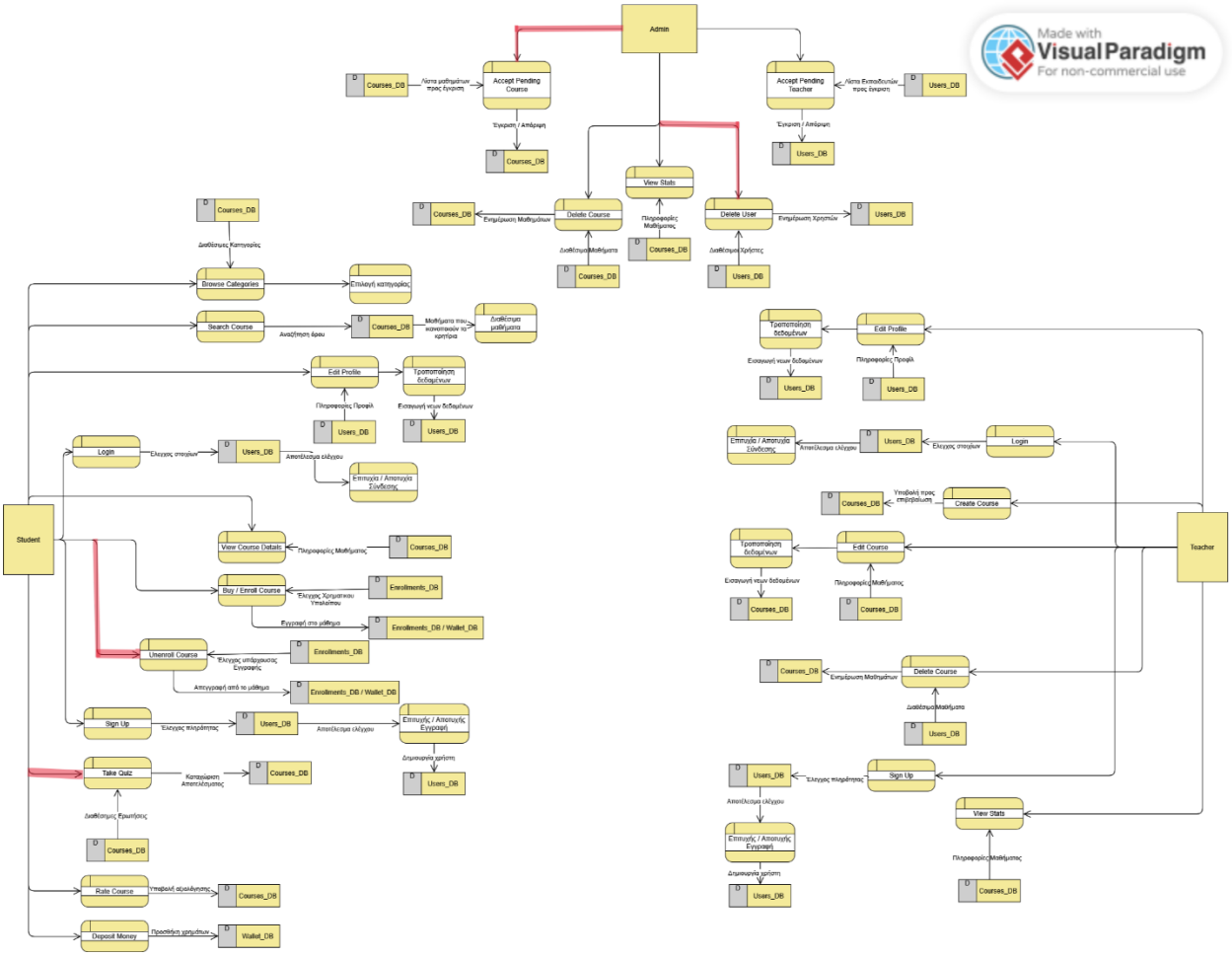
- **Register/Login**: αποστέλλεται αίτημα προς users, εκδίδεται JWT token και χρησιμοποιείται για μελλοντικές κλήσεις.
- **Create/Edit/Delete Course**: προσπελάζει την υπηρεσία courses για τη διαχείριση περιεχομένου, αποκλειστικά από χρήστες με ρόλο teacher.
- **Enroll**: αλληλεπιδρά με payment για οικονομική επιβεβαίωση και ακολούθως με enrollment για εγγραφή στο μάθημα.
- **Rate Course**: καταγράφει την αξιολόγηση του μαθητή στο μάθημα στο οποίο είναι εγγεγραμμένος.
- **Admin approvals**: διαχειρίζονται αιτήματα έγκρισης εκπαιδευτών μέσω κατάλληλων endpoints.

Οι λειτουργίες αυτές ολοκληρώθηκαν με τη βοήθεια των proxy views της υπηρεσίας myacademy, η οποία επεξεργάζεται τα tokens, εκτελεί routing και επιτυγχάνει τον συντονισμό μεταξύ των υπηρεσιών.

Η λειτουργική ροή είναι σχεδιασμένη με τρόπο που υποστηρίζει κλιμάκωση, επεκτασιμότητα και συντήρηση, και επιτρέπει στο κάθε service να εξελίσσεται ανεξάρτητα.

Με κόκκινο χρώμα στο DFD αποτυπώνονται οι **λειτουργίες που δεν υλοποιήθηκαν** στην παρούσα φάση, και αντιστοιχούν στα βασικά στοιχεία του MVP, όπως ορίστηκαν στο Κεφάλαιο 2.2. Περιλαμβάνει μεταξύ άλλων:

Microservices DFD Diagram:



Εικόνα 4.4: DFD Διάγραμμα έκδοσης Microservices

Δεν υλοποιήθηκαν οι εξής λειτουργίες:

- Delete user
- Accept Pending Course
- Unenroll Course
- Take Quiz

Delete User

Η δυνατότητα πλήρους διαγραφής χρήστη από το σύστημα δεν είναι ενδεδειγμένη τακτική σε πραγματικά συστήματα, κυρίως λόγω της σχέσης του χρήστη με άλλα δομικά δεδομένα: εγγραφές σε μαθήματα, ιστορικό συναλλαγών, αξιολογήσεις μαθημάτων κ.α. Αντί για διαγραφή από τη βάση δεδομένων, υλοποιήθηκε μηχανισμός **απενεργοποίησης λογαριασμού**, κατά τον οποίο ο χρήστης φέρει flag is_active = false, ώστε να διατηρούνται τα δεδομένα του, αλλά να αποκλείεται η πρόσβαση ή αλληλεπίδραση με την πλατφόρμα.

Accept Pending Course

Η συγκεκριμένη λειτουργία αφορά περισσότερο το εσωτερικό workflow διαχείρισης περιεχομένου, και δεν κρίθηκε απαραίτητο να υλοποιηθεί στο πρώτο στάδιο, καθώς δεν υπήρχε ρόλος πολλαπλών εκπαιδευτών που θα απαιτούσε moderation. Στο MVP, οι εκπαιδευτές έχουν ελεγχόμενη πρόσβαση και δεν υπάρχει ανάγκη για εγκρίσεις, καθώς όλα τα μαθήματα θεωρούνται εγκεκριμένα εξαρχής. Η λειτουργία θα πρέπει να προστεθεί μελλοντικά με χρήση κατάλληλου attribute status (π.χ. pending, approved, rejected).

Unenroll Course

Η απεγγραφή μαθητή από μάθημα δεν υλοποιήθηκε γιατί στο βασικό μοντέλο του MVP κάθε εγγραφή θεωρείται μόνιμη, ειδικά σε μαθήματα που συνοδεύονται από πληρωμή. Επιπλέον, η απεγγραφή εμπλέκει θέματα επιστροφής χρημάτων, επανυπολογισμού δεδομένων και διαχείρισης των συνεπαγόμενων αλλαγών σε δεδομένα, τα οποία θα απαιτούσαν μια ξεχωριστή υπηρεσία. Ως εκ τούτου, λόγω περιορισμένου χρονικού περιθωρίου επιλέχθηκε να μην υλοποιηθεί προσωρινά ώστε να διατηρηθεί η απλότητα της δομής και της εμπειρίας χρήστη.

Take Quiz

Η λειτουργία quiz απαιτεί ειδική υποδομή για τη διαχείριση ερωτήσεων, απαντήσεων, βαθμολόγησης και αποθήκευσης προόδου. Πρόκειται για ένα πλήρες υποσύστημα που απαιτεί frontend διαχείριση (quiz UI), backend business logic (validation, scoring) και αποθήκευση αποτελεσμάτων ανά χρήστη/μάθημα. Στο πλαίσιο της υλοποίησης του MVP, δόθηκε προτεραιότητα στις λειτουργίες αγοράς, εγγραφής και παρακολούθησης μαθημάτων, ενώ η υλοποίηση quiz μεταφέρεται ως επόμενη φάση για μελλοντική ανάπτυξη σε αυτόνομο microservice.

5. Συγκριτική Ανάλυση και Αξιολόγηση

5.1. Ανάλυση κοινής λειτουργικότητας στις δύο πλατφόρμες

Αν και οι δύο υλοποιήσεις της πλατφόρμας βασίστηκαν σε διαφορετικές τεχνολογικές βάσεις, μία με χρήση WordPress (μονολιθική) και μία με αρχιτεκτονική microservices, επιδιώχθηκε η υλοποίηση του ίδιου πυρήνα λειτουργιών, όπως αυτές προέκυψαν από την ανάλυση του MVP.

Το **κοινό λειτουργικό μονοπάτι** περιλαμβάνει τις βασικές διαδικασίες που είναι απαραίτητες για τη βασική εμπειρία χρήστη κάθε υλοποίησης. Η ενοποίηση αυτών των λειτουργιών στα δύο διαφορετικά μοντέλα δείχνει ότι η σχεδίαση έγινε με βάση λειτουργική ουδετερότητα, δηλαδή, ανεξαρτήτως υλοποίησης, ο χρήστης έχει παρόμοια εμπειρία και διαδρομή μέσα στην πλατφόρμα.

Στο DFD της WordPress εκδοχής, αυτές οι λειτουργίες υλοποιούνται μέσω των ενσωματωμένων modules του Tutor LMS και άλλων plugins. Στο DFD της υλοποίησης με μικροϋπηρεσίες, αντιστοιχούν σε συγκεκριμένες κλήσεις μέσω των proxy views του myacademy, οι οποίες κατευθύνουν αιτήματα στα users, courses, enrollment και payment services.

Οι διαφορές ανάμεσα στα δυο DFD διαγράματα είναι ότι στην περίπτωση των microservices έχουν προστεθεί δυο επιπλέον λειτουργίες:

- **Unenroll Course** για την απεγγραφή μαθητή από το μάθημα (Δεν υλοποιήθηκε)
- **Deposit Money** για την προσθήκη χρημάτων στο Virtual Wallet

5.2. Εικονικό πλάνο παραγωγικής μετάβασης (deployment plan)

Η επιτυχία μιας εκπαιδευτικής πλατφόρμας δεν εξαρτάται μόνο από την υλοποίησή της αλλά και από το κατά πόσο είναι παραγωγικά αξιοποιήσιμη, επεκτάσιμη και συντηρήσιμη. Στο παρόν κεφάλαιο περιγράφεται ένα **εικονικό σενάριο μετάβασης** της εφαρμογής σε πραγματικό περιβάλλον, με βάση τις ιδιαιτερότητες της κάθε προσέγγισης.

WordPress Υλοποίηση – Deployment Στρατηγική

Η υλοποίηση με WordPress στηρίζεται σε μια μονολιθική εφαρμογή, γεγονός που επιτρέπει την γρήγορη και απλή μεταφορά της σε περιβάλλον παραγωγής. Το deployment μπορεί να πραγματοποιηθεί σε shared hosting, VPS ή και σε cloud περιβάλλον (π.χ. AWS ή cPanel-based πλατφόρμες). Η βάση δεδομένων (MySQL) και ο φάκελος αρχείων WordPress μεταφέρονται μέσω migration plugin (π.χ. All-in-One WP Migration).

Τα απαραίτητα βήματα περιλαμβάνουν:

- Προετοιμασία του server (Apache/Nginx, PHP, MySQL)
- Ανέβασμα αρχείων της πλατφόρμας και επαναφορά της βάσης δεδομένων
- Εγκατάσταση SSL για ασφαλή πρόσβαση
- Ενεργοποίηση caching, CDN και backup plugins
- Παραμετροποίηση ρόλων και διασφάλιση file permissions

Πλεονέκτημα αυτής της μεθόδου είναι η γρήγορη εγκατάσταση και περιορισμένες τεχνικές απαιτήσεις. Παρ' όλα αυτά, σε περιβάλλοντα μεγάλης κλίμακας, η προσέγγιση αυτή δεν επεκτείνεται εύκολα σε πολλούς servers.

Microservices Υλοποίηση – Deployment Στρατηγική

Η μετάβαση της microservices εκδοχής απαιτεί έναν περισσότερο δομημένο μηχανισμό ανάπτυξης, αλλά αποδίδει καλύτερα σε όρους επεκτασιμότητας και ευελιξίας. Η εφαρμογή μπορεί να αναπτυχθεί σε container-based υποδομή (π.χ. Docker και Docker Compose), σε cloud providers (AWS ECS, Google Cloud Run, Azure App Services) ή σε Kubernetes (αν η κλίμακα είναι μεγαλύτερη).

Η προτεινόμενη στρατηγική παραγωγικής μετάβασης περιλαμβάνει:

- Δημιουργία Docker images για κάθε microservice (users, courses, enrollment, payment, myacademy)
- Ορισμός .env αρχείων με παραμέτρους ασφαλείας και σύνδεσης
- Docker Compose αρχείο που χρησιμοποιεί όλα τα services, περιλαμβανομένων των βάσεων δεδομένων (PostgreSQL, SQLite ή Mongo)
- Reverse proxy (Nginx) με SSL termination
- Σύστημα για logging (π.χ. Loki ή ELK), monitoring (Prometheus) και health checks
- Έλεγχος scaling (π.χ. με horizontal auto-scaling) και fault-tolerance

Παρότι αυτή η μέθοδος απαιτεί περισσότερη τεχνική κατάρτιση, προσφέρει σημαντικά πλεονεκτήματα σε πλατφόρμες που αναμένεται να εξελιχθούν, να υποστηρίξουν concurrency, ή να δεχθούν πολλαπλές αναβαθμίσεις ανεξαρτήτως service.

Η τελική επιλογή deployment strategy εξαρτάται από το προφίλ της ομάδας ανάπτυξης, τον προϋπολογισμό, τις απαιτήσεις κλιμάκωσης και την αναμενόμενη κυκλοφορία της εφαρμογής. Για μια

αρχική κυκλοφορία, το WordPress αποτελεί μια γρήγορη λύση· ωστόσο, σε σενάρια αυξημένης χρήσης, η αρχιτεκτονική microservices υπερέχει ως προς την ανθεκτικότητα και την προσαρμοστικότητα.

5.3. Ανάλυση και αξιολόγηση μέσω KPI (Key Performance Indicators)

5.3.1. Δείκτες KPI και η σημασία τους

Οι Δείκτες Απόδοσης (Performance Indicators) αποτελούν θεμελιώδες εργαλείο στη σύγχρονη διοίκηση οργανισμών, καθώς επιτρέπουν την αποτύπωση και αξιολόγηση της προόδου προς την επίτευξη των στρατηγικών και επιχειρησιακών στόχων. Η έννοια των Key Performance Indicators (KPIs), δηλαδή των «Βασικών Δεικτών Απόδοσης», αναφέρεται σε ένα υποσύνολο αυτών των δεικτών, οι οποίοι εστιάζουν στους κρίσιμους παράγοντες επιτυχίας ενός οργανισμού.

Σύμφωνα με τον David Parmenter [13], τα KPIs διακρίνονται από τους υπόλοιπους δείκτες λόγω της ικανότητάς τους να επηρεάζουν άμεσα τη συμπεριφορά των εργαζομένων και την απόδοση των μονάδων. Ένας καλά επιλεγμένος KPI δεν είναι απλώς μια μέτρηση είναι ένα μέσο βελτίωσης, που ενισχύει την στρατηγική και την λειτουργία.

Είναι σημαντικό να γίνει διάκριση μεταξύ των KPIs και των Κρίσιμων Παραγόντων Επιτυχίας (Critical Success Factors - CSFs). Οι CSFs αντιπροσωπεύουν τις βασικές προϋποθέσεις για την επιτυχία ενός οργανισμού και συχνά προκύπτουν από τη στρατηγική ανάλυση. Οι KPIs, από την άλλη πλευρά, είναι μετρήσεις που καθιστούν δυνατή την παρακολούθηση αυτών των παραγόντων στην πράξη.

Ένα από τα βασικά επιχειρήματα του Parmenter είναι ότι η παρανόηση του τι αποτελεί πραγματικά KPI οδηγεί συχνά στην επιλογή δείκτων που δεν είναι κατάλληλοι ή χρήσιμοι. Οι περισσότεροι οργανισμοί χρησιμοποιούν μια πληθώρα μετρήσεων, εκ των οποίων μόνο λίγες είναι πραγματικοί KPIs. Το πρώτο βήμα προς τη σωστή επιλογή είναι η κατανόηση αυτής της διαφοροποίησης και η αναγνώριση του μικρού αλλά κρίσιμου συνόλου δεικτών που οδηγούν στην επιθυμητή αλλαγή.

5.3.2. Χαρακτηριστικά Αποτελεσματικών KPIs

Στο έργο του David Parmenter, επισημαίνονται ορισμένα κρίσιμα χαρακτηριστικά που διακρίνουν έναν αποτελεσματικό KPI από τις απλές μετρήσεις. Η κατανόηση αυτών των γνωρισμάτων είναι απαραίτητη για τη δημιουργία ενός αξιόπιστου και χρήσιμου πλαισίου παρακολούθησης της οργανωσιακής απόδοσης.

1. **Μη χρηματοοικονομική φύση:** Οι πραγματικοί KPIs δεν βασίζονται μονο σε οικονομικά δεδομένα, καθώς αυτά είναι συχνά αποτελέσματα παρελθοντικών ενεργειών. Αντίθετα, εστιάζουν σε λειτουργικές δραστηριότητες που μπορούν να επηρεαστούν άμεσα από τους εργαζομένους.

2. **Συχνή μέτρηση:** Η συχνότητα είναι ουσιώδης. Οι KPIs πρέπει να ενημερώνονται τακτικά ώστε να προσφέρουν έγκαιρη πληροφόρηση και να επιτρέπουν διορθωτικές κινήσεις σε πραγματικό χρόνο.
3. **Κατανόηση και δράση από την ομάδα:** Οι δείκτες πρέπει να είναι απλοί και κατανοητοί από όλα τα μέλη του οργανισμού. Αν δεν είναι σαφές πώς συνδέονται με την καθημερινή εργασία ή δεν οδηγούν σε άμεση δράση, τότε δεν λειτουργούν ως πραγματικοί KPIs.
4. **Ανάθεση ευθύνης σε άτομα ή ομάδες:** Κάθε KPI πρέπει να συνδέεται με συγκεκριμένη λειτουργική μονάδα ή υπεύθυνο, ώστε να διασφαλίζεται η λογοδοσία και η ενεργή παρακολούθηση.
5. **Εστίαση σε κρίσιμες δραστηριότητες:** Οι KPIs πρέπει να σχετίζονται με διαδικασίες που υποστηρίζουν τους Κρίσιμους Παράγοντες Επιτυχίας και να στοχεύουν στη συνεχή βελτίωση αυτών των τομέων.
6. **Συμπεριφορική επίδραση:** Ένας πραγματικός KPI πρέπει να επηρεάζει θετικά τη συμπεριφορά του προσωπικού και να ενισχύει την επιθυμητή απόδοση.

Η επιλογή των KPIs πρέπει να γίνεται με εξαιρετική προσοχή, καθώς η λανθασμένη μέτρηση οδηγεί συχνά σε εσφαλμένες αποφάσεις και στρατηγικό αποπροσανατολισμό.

5.3.3. Στάδια Επιλογής KPIs

Η διαδικασία επιλογής των Key Performance Indicators είναι πολυσταδιακή και απαιτεί τη συμμετοχή πολλών επιπέδων του οργανισμού. Στόχος είναι η εντοπισμένη ανάπτυξη KPIs που αντανακλούν τους κρίσιμους παράγοντες επιτυχίας (CSFs) και ενισχύουν τη στρατηγική του οργανισμού. Τα βασικά στάδια περιλαμβάνουν:

1. **Διαμόρφωση Ομάδας Υλοποίησης (KPI Team):** Η διαδικασία ξεκινά με τη συγκρότηση μιας διατμηματικής ομάδας έργου, η οποία αναλαμβάνει την ευθύνη για τον σχεδιασμό και την υλοποίηση του πλαισίου KPIs. Η ομάδα πρέπει να περιλαμβάνει άτομα από διαφορετικά ιεραρχικά επίπεδα και επιχειρησιακές λειτουργίες.
2. **Αναγνώριση Κρίσιμων Παραγόντων Επιτυχίας (CSFs):** Με βάση τη στρατηγική και τις βασικές λειτουργίες του οργανισμού, η ομάδα εντοπίζει τους CSFs που αποτελούν τους βασικούς άξονες στους οποίους πρέπει να εστιάζουν οι δείκτες απόδοσης.
3. **Ανάλυση Υφιστάμενων Μετρήσεων:** Καταγράφονται και αξιολογούνται όλοι οι δείκτες που χρησιμοποιούνται ήδη στον οργανισμό, με σκοπό να διαπιστωθεί ποιοι έχουν πραγματική επιχειρησιακή αξία και ποιοι χρειάζονται αναθεώρηση ή απόρριψη.
4. **Συνεντεύξεις και Ερωτηματολόγια:** Μέσω ατομικών συνεντεύξεων και ερωτηματολογίων, η ομάδα KPI συλλέγει πληροφορίες από όλα τα επίπεδα του οργανισμού σχετικά με τις ανάγκες πληροφόρησης, τις προκλήσεις και τις αντιλήψεις γύρω από την απόδοση.
5. **Διοργάνωση Συναντήσεων (KPI Workshops):** Οι συναντήσεις αποτελούν το κεντρικό εργαλείο δημιουργικής αναζήτησης KPIs. Συμμετέχουν εκπρόσωποι όλων των κρίσιμων μονάδων και μέσω συζήτησης και ανταλλαγής απόψεων καταλήγουν σε ένα σύνολο υποψήφιων KPIs για κάθε CSF.
6. **Κατηγοριοποίηση και Ιεράρχηση των KPIs:** Οι δείκτες μετατρέπονται σε KPIs. Δίνεται προτεραιότητα σε όσους δείκτες έχουν ισχυρή επίδραση στην απόδοση και ανταποκρίνονται στα χαρακτηριστικά που προαναφέρθηκαν.
7. **Ανάθεση Ευθυνών και Πλαισίου Παρακολούθησης:** Για κάθε KPI ορίζονται υπεύθυνοι, μηχανισμοί μέτρησης, συχνότητα αναφορών και απαιτήσεις πληροφορίας.
8. **Πιλοτική Εφαρμογή και Ανατροφοδότηση:** Το τελικό σύνολο KPIs δοκιμάζεται σε πιλοτικό περιβάλλον ώστε να εντοπιστούν αδυναμίες και να γίνουν οι απαραίτητες προσαρμογές.

Αυτό το μοντέλο ενσωματώνει τη συμμετοχή των εργαζομένων και τη συνεχή ανατροφοδότηση, διασφαλίζοντας ότι οι KPIs δεν είναι απλώς διαχειριστικοί δείκτες, αλλά εργαλεία ενεργής υποστήριξης της στρατηγικής και καθημερινής βελτίωσης.

5.3.4. Τεκμηρίωση Επιλογής KPIs

Η επιλογή των δεικτών απόδοσης που παρουσιάζονται στη συνέχεια της διπλωματικής δεν προέκυψε μέσα από συλλογικές διαδικασίες ή δομημένες ομάδες εργασίας, όπως συνιστούν αρκετά πρότυπα ανάπτυξης KPIs (π.χ. workshops, cross-functional teams). Αντιθέτως, αποτέλεσε αποκλειστική εργασία ενός ατόμου, γεγονός που επέβαλε ιδιαίτερη προσοχή, αυστηρή τεκμηρίωση και προσήλωση στην επιλογή των δεικτών.

Οι κατηγορίες KPI που υιοθετήθηκαν (όπως Απόδοση, Διαθεσιμότητα, Ασφάλεια, Κόστος, Χρήστες, Μάρκετινγκ, Συντήρηση Λογισμικού, Κλιμάκωση, Ευελιξία) βασίζονται:

Δομημένη αιτιολόγηση ανά κατηγορία

- **Απόδοση (Performance):** Οι δείκτες όπως *Page Load Time*, *Time to First Byte (TTFB)* και *Concurrent Users Handling* είναι standard στον χώρο της ανάπτυξης web εφαρμογών και αναγνωρίζονται από διεθνή frameworks όπως το Google Lighthouse ή το W3C ως βασικά metrics εμπειρίας χρήστη. [14]
- **Ασφάλεια (Security):** Δείκτες όπως *Vulnerabilities Detected* και *Data Breach Risk* εναρμονίζονται με πρακτικές ασφαλείας όπως το OWASP Top 10. [15]
- **Κόστος (Cost):** Το *Monthly Infrastructure Cost* και το *3rd Party Service Cost* θεωρούνται πλέον αναπόσπαστα στοιχεία της βιωσιμότητας κάθε cloud-based λύσης [16]
- **Scalability & Business Agility:** Η επιλογή KPIs όπως *Infrastructure Elasticity* και *Time to Market* ευθυγραμμίζεται με best practices σε DevOps περιβάλλοντα και agile ανάπτυξη [17]

Αντικειμενική ανάγκη πολυδιάστατης μέτρησης

Η φύση της πλατφόρμας απαιτούσε μετρήσεις που καλύπτουν και τεχνικά χαρακτηριστικά (π.χ. απόδοση, συντήρηση) και επιχειρησιακά (π.χ. churn rate, revenue per user). Η ολιστική αυτή προσέγγιση διασφαλίζει την αξιολόγηση της πλατφόρμας όχι μόνο ως τεχνικό έργο αλλά και ως επιχειρησιακό προϊόν.

Περιορισμοί ατομικής εργασίας

Η απουσία ομάδας σήμαινε ότι δεν υπήρχε δυνατότητα διασταύρωσης απόψεων ή εμπλουτισμού των KPIs μέσα από εμπειρική γνώση διαφορετικών stakeholders. Ωστόσο, αυτό αντισταθμίστηκε από την εμπειρική παρατήρηση κατά τη διάρκεια της υλοποίησης.

5.3.5. Ανάλυση Δεικτών KPI

1. Page Load Time

Ο δείκτης *Page Load Time* αναφέρεται στον συνολικό χρόνο που απαιτείται για να φορτωθεί πλήρως μια ιστοσελίδα στο πρόγραμμα περιήγησης του χρήστη, από τη στιγμή που θα ζητηθεί μέχρι την πλήρη απόδοση όλων των στοιχείων (HTML, CSS, JavaScript, εικόνες, γραμματοσειρές). Η μέτρηση γίνεται είτε μέσω εργαλείων frontend όπως το Google Lighthouse, το WebPageTest ή το Chrome DevTools, είτε με real user monitoring (RUM) σε παραγωγικό περιβάλλον. Για τον υπολογισμό του δείκτη λαμβάνεται μέσος όρος από πολλαπλές δοκιμές σε διαφορετικές συσκευές, συνδέσεις και γεωγραφικές τοποθεσίες. Η τιμή αποτυπώνει όχι μόνο την απόδοση του frontend, αλλά και παράγοντες όπως το CDN, το caching, και τη βαρύτητα των πόρων της σελίδας.

2. Server Response Time

Ο *Server Response Time* (ή χρόνος απόκρισης server) είναι ο χρόνος που μεσολαβεί από τη στιγμή που γίνεται ένα αίτημα HTTP από τον browser έως τη στιγμή που ο server αρχίζει να επιστρέφει δεδομένα στον client. Περιλαμβάνει τον χρόνο που απαιτείται για την αναζήτηση, επεξεργασία και προετοιμασία της απάντησης από τον server. Ο υπολογισμός του γίνεται με εργαλεία παρακολούθησης όπως το Lighthouse, New Relic ή custom backend logging, και μετριέται σε milliseconds (ms). Για κάθε endpoint πραγματοποιείται μέτρηση σε διάφορες χρονικές στιγμές και συνθήκες, ενώ το τελικό αποτέλεσμα είναι ο μέσος χρόνος απόκρισης, π.χ. σε ένα 24ωρο ή σε συγκεκριμένο χρονικό διάστημα. Ο δείκτης αυτός βοηθά στην αναγνώριση προβλημάτων απόδοσης στο backend ή στη βάση δεδομένων.

3. Time to First Byte (TTFB)

Το *Time to First Byte (TTFB)* είναι ο χρόνος που απαιτείται από τη στιγμή που ο browser του χρήστη στέλνει ένα αίτημα HTTP στον server μέχρι τη στιγμή που λαμβάνει το πρώτο byte από την απάντηση. Ουσιαστικά αποτελεί άμεσο δείκτη της απόδοσης του web server και της αρχικής δικτυακής καθυστέρησης. Ο υπολογισμός του γίνεται μέσω εργαλείων όπως το Chrome DevTools, GTmetrix ή μέσω ενσωματωμένων εργαλείων παρακολούθησης CDN (π.χ. Cloudflare). Στη μέτρηση περιλαμβάνονται: ο χρόνος DNS lookup, ο χρόνος TCP handshake, ο χρόνος SSL negotiation (αν υπάρχει), και τελικά ο χρόνος επεξεργασίας από τον server. Το TTFB εκφράζεται σε milliseconds και είναι κρίσιμος δείκτης για την εμπειρία χρήστη, καθώς μεγάλοι χρόνοι συνεπάγονται καθυστερημένο rendering της σελίδας.

4. Time to Interactive (TTI)

Το *Time to Interactive (TTI)* μετρά τον χρόνο που απαιτείται ώστε μια ιστοσελίδα να είναι πλήρως λειτουργική και διαδραστική, δηλαδή να μπορεί ο χρήστης να αλληλεπιδράσει χωρίς καθυστερήσεις ή προβλήματα απόκρισης. Ο υπολογισμός του βασίζεται σε μετρήσεις από εργαλεία όπως το Google Lighthouse και το Web Vitals API, τα οποία ανιχνεύουν πότε το βασικό περιεχόμενο έχει φορτωθεί, όλα τα βασικά scripts έχουν εκτελεστεί, και το thread του

browser είναι επαρκώς ελεύθερο ώστε να ανταποκριθεί σε εντολές χρήστη. Συμπεριλαμβάνει το χρόνο parsing και εκτέλεσης JavaScript, rendering του DOM, και άλλων εργασιών που μπλοκάρουν τη διαδραστικότητα. Το TTI είναι κρίσιμο για εφαρμογές με πλούσια frontend εμπειρία και μετριέται σε milliseconds ή δευτερόλεπτα.

5. Concurrent Users Handling

Το *Concurrent Users Handling* αναφέρεται στον μέγιστο αριθμό χρηστών που μπορούν να χρησιμοποιούν την εφαρμογή ή την πλατφόρμα ταυτόχρονα, χωρίς σημαντική υποβάθμιση της απόδοσης. Ο δείκτης αυτός αξιολογείται μέσω stress testing και load testing εργαλείων όπως το JMeter, Gatling ή k6. Κατά τη διάρκεια των δοκιμών, αυξάνεται σταδιακά ο αριθμός των χρηστών που εκτελούν τυπικές ενέργειες στο σύστημα (login, αναζήτηση, φόρτωση σελίδων), και καταγράφονται κρίσιμα metrics όπως χρόνοι απόκρισης, ρυθμός σφαλμάτων και χρήση πόρων. Η τελική τιμή αντιστοιχεί στο σημείο όπου παρατηρείται πτώση στην απόδοση ή υπέρβαση αποδεκτών ορίων latency. Ο δείκτης βοηθά στην αξιολόγηση της επεκτασιμότητας και της υποδομής, και χρησιμοποιείται για τη βελτίωση του συστήματος παραγωγής.

6. Uptime & Availability

Το KPI *Uptime & Availability* μετρά το ποσοστό του χρόνου κατά τον οποίο ένα σύστημα, υπηρεσία ή εφαρμογή είναι διαθέσιμο και λειτουργεί σωστά για τους χρήστες, χωρίς διακοπές ή αστοχίες. Υπολογίζεται ως το ποσοστό του συνολικού χρόνου λειτουργίας σε σχέση με το σύνολο του χρονικού διαστήματος παρακολούθησης (π.χ. ανά μήνα ή έτος). Ο τύπος είναι:

Availability(%) = (Total Time – Downtime) / Total Time × 100

Η παρακολούθηση γίνεται με εργαλεία όπως Pingdom, Uptime Robot, ή custom health checks που στέλνουν περιοδικά αιτήματα στον server και καταγράφουν τυχόν αποτυχίες. Ένα uptime της τάξης του 99.9% σημαίνει περίπου 43 λεπτά downtime τον μήνα. Ο δείκτης είναι κρίσιμος για SLA (Service Level Agreements), καθώς αντικατοπτρίζει τη σταθερότητα και αξιοπιστία της υποδομής.

7. Error Rate

Το *Error Rate* αναφέρεται στο ποσοστό των αιτημάτων προς την εφαρμογή ή τον server που καταλήγουν σε σφάλμα, σε σχέση με το σύνολο των αιτημάτων που πραγματοποιούνται σε ένα συγκεκριμένο χρονικό διάστημα. Υπολογίζεται με τον τύπο:

Error Rate (%) = (Number of Failed Requests / Total Requests) × 100

Οι αποτυχημένες αιτήσεις μπορεί να περιλαμβάνουν HTTP σφάλματα (π.χ. 4xx, 5xx), exceptions στο backend, ή αποτυχημένες λειτουργίες σε client-side εφαρμογές. Η μέτρηση γίνεται μέσω εργαλείων όπως Sentry, New Relic, Datadog ή custom application logging. Ο δείκτης είναι καθοριστικός για την ποιότητα της εμπειρίας χρήστη και την υγεία της εφαρμογής, καθώς

αυξημένο error rate υποδηλώνει λειτουργικά ή αρχιτεκτονικά προβλήματα που πρέπει να διερευνηθούν άμεσα.

8. Downtime per Month

Το *Downtime per Month* μετρά τον συνολικό χρόνο (σε λεπτά ή ώρες) κατά τον οποίο η υπηρεσία ή το σύστημα δεν ήταν διαθέσιμο εντός ενός ημερολογιακού μήνα. Η έννοια της «μη διαθεσιμότητας» περιλαμβάνει τόσο πλήρεις διακοπές λειτουργίας όσο και περιόδους κατά τις οποίες το σύστημα ήταν τεχνικά προσβάσιμο αλλά ουσιαστικά μη λειτουργικό (π.χ. λόγω πολύ υψηλής καθυστέρησης ή συνεχών σφαλμάτων). Ο υπολογισμός βασίζεται σε αρχεία παρακολούθησης από uptime monitoring εργαλεία, τα οποία ανιχνεύουν και καταγράφουν κάθε downtime event, με χρονική σήμανση έναρξης και λήξης. Το άθροισμα αυτών των χρονικών διαστημάτων ανά μήνα αποτελεί την τιμή του δείκτη. Ο *Downtime per Month* είναι κρίσιμος για οργανισμούς που προσφέρουν 24/7 υπηρεσίες και συνδέεται άμεσα με SLA συμμόρφωση και επιχειρησιακό ρίσκο.

9. Backup & Recovery Time

Το *Backup & Recovery Time* αναφέρεται στον συνολικό χρόνο που απαιτείται για την εκτέλεση ενός πλήρους αντιγράφου ασφαλείας (backup) και την επαναφορά (recovery) του συστήματος σε λειτουργική κατάσταση μετά από αποτυχία ή απώλεια δεδομένων. Ο υπολογισμός αυτού του KPI περιλαμβάνει δύο διακριτές φάσεις:

(α) τον χρόνο δημιουργίας και αποθήκευσης ενός πλήρους αντιγράφου ασφαλείας, και (β) τον χρόνο που απαιτείται για να επανακτηθούν τα δεδομένα και να επανεκκινηθούν οι κρίσιμες υπηρεσίες.

Οι μετρήσεις πραγματοποιούνται σε ελεγχόμενα δοκιμαστικά σενάρια ή μέσω παρακολούθησης πραγματικών περιστατικών. Εργαλεία όπως Veeam, AWS Backup, Azure Recovery Services, ή scripts αυτοματοποιημένων διαδικασιών καταγράφουν αναλυτικά αυτά τα χρονικά διαστήματα. Η αξιολόγηση του δείκτη βοηθά στον σχεδιασμό στρατηγικών business continuity και DR (Disaster Recovery), και αποτυπώνει την επιχειρησιακή ετοιμότητα του οργανισμού.

10. Vulnerabilities Detected

Το *Vulnerabilities Detected* μετρά τον αριθμό των ευπαθειών (vulnerabilities) που εντοπίζονται σε μια εφαρμογή, σύστημα ή υποδομή κατά τη διάρκεια ελέγχων ασφαλείας. Οι ευπάθειες μπορεί να αφορούν αδυναμίες στον πηγαίο κώδικα, ανασφαλείς βιβλιοθήκες, λανθασμένες ρυθμίσεις server, ή ελλείψεις στην προστασία δεδομένων. Η μέτρηση πραγματοποιείται μέσω εργαλείων στατικής και δυναμικής ανάλυσης, όπως το OWASP ZAP, Nessus, SonarQube, Snyk ή Burp Suite. Οι σαρώσεις μπορούν να είναι περιοδικές ή να εκτελούνται ως μέρος μιας CI/CD ροής. Ο δείκτης αποτυπώνει την επίπτωση των πρακτικών ασφάλειας που εφαρμόζονται και λειτουργεί

ως προειδοποιητικός μηχανισμός για πιθανούς κινδύνους, βοηθώντας στον προγραμματισμό διορθώσεων (patching) και την ενίσχυση της ασφάλειας σε βάθος.

11. Number of Security Incidents

Το *Number of Security Incidents* καταγράφει τον αριθμό επιβεβαιωμένων περιστατικών ασφαλείας που έχουν συμβεί σε μια χρονική περίοδο, όπως προσπάθειες εισβολής, παραβιάσεις λογαριασμών, επιθέσεις DDoS, αποτυχημένες προσπάθειες αυθεντικοποίησης, ή διαρροές δεδομένων. Η καταγραφή βασίζεται είτε σε αυτοματοποιημένα συστήματα παρακολούθησης (SIEM, IDS/IPS) όπως Splunk, Wazuh ή Snort, είτε σε αναφορές από διαχειριστές συστήματος και ομάδες ασφάλειας. Κάθε περιστατικό αξιολογείται ως προς τη σοβαρότητά του (π.χ. low, medium, critical) και τεκμηριώνεται. Ο δείκτης παρέχει μια συνολική εικόνα της «εκτεθειμένης επιφάνειας» της πλατφόρμας και αξιοποιείται για την αξιολόγηση της αποτελεσματικότητας των μέτρων προστασίας και των διαδικασιών απόκρισης σε περιστατικά (incident response).

12. Encryption Level

Το *Encryption Level* αξιολογεί την ποιότητα και το βάθος των τεχνικών κρυπτογράφησης που εφαρμόζονται στα δεδομένα και στις επικοινωνίες μιας εφαρμογής ή υποδομής. Περιλαμβάνει τόσο την κρυπτογράφηση εν κινήσει (*encryption in transit*) όσο και την κρυπτογράφηση σε αποθήκευση (*encryption at rest*). Η μέτρηση γίνεται με βάση το είδος και τη δύναμη του χρησιμοποιούμενου αλγορίθμου (π.χ. AES-256, RSA-2048), τη συμμόρφωση με πρότυπα (π.χ. TLS 1.2 ή 1.3), και την ύπαρξη διαδικασιών διαχείρισης κλειδιών (key rotation, secure storage). Ο δείκτης δεν είναι αυστηρά ποσοτικός, αλλά αποτιμάται μέσω ελέγχων ασφάλειας, εργαλείων αξιολόγησης (π.χ. SSL Labs), και αναφορών συμμόρφωσης. Ένας υψηλός βαθμός κρυπτογράφησης μειώνει την πιθανότητα παραβίασης δεδομένων και ενισχύει την εμπιστοσύνη χρηστών και συνεργατών.

13. Data Breach Risk

Το *Data Breach Risk* αποτιμά την πιθανότητα να προκύψει διαρροή ή μη εξουσιοδοτημένη πρόσβαση σε ευαίσθητα δεδομένα, όπως προσωπικές πληροφορίες χρηστών, οικονομικά στοιχεία ή διαπιστευτήρια. Ο δείκτης αυτός υπολογίζεται αξιολογώντας έναν συνδυασμό παραγόντων: την ύπαρξη γνωστών ευπαθειών, το επίπεδο κρυπτογράφησης, την πολυπλοκότητα της αυθεντικοποίησης, τη συμμόρφωση με πρότυπα ασφαλείας (π.χ. GDPR, ISO 27001), και το ιστορικό προηγούμενων περιστατικών. Συχνά υπολογίζεται μέσω risk assessment εργαλείων ή frameworks, όπως το NIST Risk Management Framework ή το CVSS (Common Vulnerability Scoring System). Αν και δεν είναι απόλυτα ποσοτικός δείκτης, η ποιοτική ή ημι-ποσοτική εκτίμηση του βοηθά στη λήψη αποφάσεων για ενίσχυση των πολιτικών ασφαλείας, την ιεράρχηση δράσεων και τη μείωση της έκθεσης σε κανονιστικές κυρώσεις και απώλεια εμπιστοσύνης.

14. Initial Development Cost

Το *Initial Development Cost* αναφέρεται στο συνολικό κόστος που απαιτείται για τη σχεδίαση, ανάπτυξη και παράδοση της πρώτης λειτουργικής έκδοσης της εφαρμογής ή του συστήματος. Ο υπολογισμός του γίνεται με βάση την εκτιμώμενη διάρκεια ανάπτυξης (σε ανθρωπομήνες ή εβδομάδες), σε συνδυασμό με το κόστος εργατικού δυναμικού (μέσοι μισθοί προγραμματιστών και σχετικών ρόλων με βάση δεδομένα αγοράς). Επιπλέον, λαμβάνονται υπόψη το κόστος εξοπλισμού (υπολογιστές, λογισμικά αδειών, IDEs), πιθανές εξωτερικές υπηρεσίες (outsourcing, συμβουλευτική), και έξοδα αρχικής υποδομής (cloud, domain, dev/test περιβάλλοντα). Η τιμή του KPI εκφράζεται σε ευρώ ή άλλο νόμισμα και παρέχει καθοριστικό μέτρο για τη χρηματοοικονομική αποδοτικότητα του έργου, καθώς και για benchmarking μελλοντικών αναπτύξεων.

15. Monthly Infrastructure Cost

Το *Monthly Infrastructure Cost* καταγράφει το συνολικό κόστος που σχετίζεται με τη λειτουργία και υποστήριξη της τεχνικής υποδομής της εφαρμογής σε μηνιαία βάση. Περιλαμβάνει χρεώσεις από cloud υπηρεσίες (π.χ. AWS, Azure, Google Cloud), όπως compute (VMs, containers), storage (βάσεις δεδομένων, backups), networking (bandwidth, CDN), και πρόσθετες υπηρεσίες (monitoring, logging, load balancers). Επιπλέον, μπορεί να περιλαμβάνει κόστος από dedicated servers, υπηρεσίες DNS, email ή firewall, ανάλογα με την αρχιτεκτονική. Η μέτρηση γίνεται συνήθως μέσω των αναλυτικών τιμολογίων των παρόχων ή με dashboards που παρέχουν συγκεντρωτικά κόστη ανά κατηγορία. Ο δείκτης είναι κρίσιμος για την παρακολούθηση της βιωσιμότητας του έργου, την πρόβλεψη εξόδων και τον έλεγχο υπερβάσεων προϋπολογισμού.

16. Maintenance & Support Cost

Το *Maintenance & Support Cost* αντιπροσωπεύει το συνολικό κόστος που απαιτείται για τη συνεχή συντήρηση και υποστήριξη μιας εφαρμογής ή πλατφόρμας μετά την αρχική της κυκλοφορία. Περιλαμβάνει την εργασία προγραμματιστών και τεχνικών για επιδιορθώσεις σφαλμάτων, εφαρμογή ενημερώσεων ασφαλείας, βελτιώσεις λειτουργιών, διαχείριση βάσεων δεδομένων, καθώς και την υποστήριξη τελικών χρηστών (helpdesk, ticketing). Ο υπολογισμός του δείκτη γίνεται με βάση το κόστος εργατοώρας του σχετικού προσωπικού, ενδεχόμενες συνδρομές σε εργαλεία monitoring ή bug tracking, και πρόσθετα έξοδα (όπως σύμβουλοι ή υποστήριξη τρίτων). Το κόστος αυτό παρακολουθείται σε τακτική βάση (μηνιαία ή τριμηνιαία) και παρέχει σημαντική εικόνα για τη βιωσιμότητα του έργου, καθώς ένα δυσανάλογα υψηλό ποσό μπορεί να υποδηλώνει προβλήματα στην ποιότητα του αρχικού κώδικα ή στην αρχιτεκτονική.

17. 3rd Party Service Costs

Το *3rd Party Service Costs* αναφέρεται στο μηνιαίο ή ετήσιο κόστος που σχετίζεται με την ενσωμάτωση και χρήση υπηρεσιών τρίτων στην πλατφόρμα ή την εφαρμογή. Περιλαμβάνει άδειες χρήσης APIs (π.χ. υπηρεσίες πληρωμών, αποστολής email, αναλύσεων), συνδρομές σε πλατφόρμες SaaS (όπως CMS, authentication providers, AI APIs), καθώς και κόστη plugin,

libraries ή modules που δεν είναι δωρεάν. Ο υπολογισμός γίνεται καταγράφοντας όλες τις εξωτερικές εξαρτήσεις που επιφέρουν οικονομική επιβάρυνση και συνυπολογίζοντας είτε σταθερές τιμές είτε μεταβλητές χρεώσεις ανάλογα με τη χρήση (π.χ. κόστος ανά 1.000 requests). Ο δείκτης είναι σημαντικός για την εκτίμηση του πραγματικού κόστους λειτουργίας της εφαρμογής, την επιλογή συνεργατών με βάση την απόδοση προς κόστος, αλλά και για στρατηγικές απεξάρτησης από εξωτερικούς παρόχους όπου χρειάζεται.

18. User Churn Rate

Το *User Churn Rate* μετρά το ποσοστό των χρηστών που σταματούν να χρησιμοποιούν την εφαρμογή ή διακόπτουν τη συνδρομή τους μέσα σε ένα συγκεκριμένο χρονικό διάστημα. Υπολογίζεται με τον τύπο:

Churn Rate (%) = (Χρήστες που αποχώρησαν / Συνολικοί ενεργοί χρήστες στην αρχή της περιόδου) × 100

Η παρακολούθηση γίνεται μέσα από το σύστημα διαχείρισης χρηστών (user database) ή μέσω εργαλείων analytics (π.χ. Mixpanel, Amplitude, Firebase). Ο δείκτης αυτός είναι κρίσιμος για προϊόντα με μοντέλο συνδρομής ή συνεχή χρήση, καθώς υποδηλώνει την ικανοποίηση των χρηστών, την απόδοση των υπηρεσιών και πιθανές αδυναμίες στο onboarding ή στη διατήρηση engagement. Η ανάλυση του churn μπορεί να γίνει σε διαφορετικά επίπεδα (ημερήσιο, μηνιαίο, ετήσιο) και συχνά συνοδεύεται από cohort analysis για τον εντοπισμό μοτίβων εγκατάλειψης.

19. Active Users per Month (MAU)

Το *Monthly Active Users (MAU)* μετρά τον συνολικό αριθμό μοναδικών χρηστών που αλληλεπίδρασαν ενεργά με την εφαρμογή ή την πλατφόρμα τουλάχιστον μία φορά μέσα σε έναν ημερολογιακό μήνα. «Ενεργή χρήση» ορίζεται συνήθως ως είσοδος στην πλατφόρμα, συμμετοχή σε βασικές ενέργειες (π.χ. παρακολούθηση περιεχομένου, αγορές, χρήση βασικών λειτουργιών), και όχι απλώς άνοιγμα της εφαρμογής. Τα δεδομένα εξάγονται μέσω συστημάτων analytics ή απευθείας από τα logs χρήστη. Το MAU χρησιμοποιείται ευρέως για την αποτύπωση της απήχησης μιας πλατφόρμας, την παρακολούθηση ανάπτυξης κοινού, και τη σύγκριση με άλλους δείκτες όπως το Daily Active Users (DAU) ή τον Churn Rate. Ο δείκτης είναι επίσης σημαντικός για την αξιολόγηση της βιωσιμότητας του έργου και την πρόβλεψη εσόδων σε freemium ή συνδρομητικά μοντέλα.

20. Customer Satisfaction Score (CSAT)

Το Customer Satisfaction Score (CSAT) είναι ένας δείκτης που μετρά την ικανοποίηση των χρηστών ή πελατών από μια συγκεκριμένη εμπειρία, υπηρεσία ή λειτουργία της εφαρμογής. Συλλέγεται συνήθως μέσω σύντομων ερωτηματολογίων που εμφανίζονται μετά από μια ενέργεια (π.χ. υποβολή αιτήματος υποστήριξης, ολοκλήρωση αγοράς ή μαθήματος) και ζητούν από τον χρήστη να βαθμολογήσει την εμπειρία του σε κλίμακα (π.χ. 1–5 ή 1–10). Ο υπολογισμός γίνεται ως εξής:

CSAT (%) = (Αριθμός θετικών απαντήσεων / Συνολικός αριθμός απαντήσεων) × 100

Θετικές θεωρούνται συνήθως οι απαντήσεις στην ανώτερη κλίμακα (π.χ. 4 και 5 σε 5βάθμια κλίμακα). Ο δείκτης χρησιμοποιείται για την αξιολόγηση της ποιότητας υποστήριξης, της χρηστικότητας ή της συνολικής εμπειρίας, και αποτελεί βασικό εργαλείο για τη λήψη feedback και τη βελτίωση προϊόντων και υπηρεσιών.

21. Course Completion Rate

Το *Course Completion Rate* μετρά το ποσοστό των χρηστών που ολοκληρώνουν με επιτυχία ένα course ή εκπαιδευτική ενότητα σε σχέση με όσους το ξεκίνησαν. Είναι κρίσιμος δείκτης για πλατφόρμες e-learning, καθώς αποτυπώνει τόσο την ποιότητα και ελκυστικότητα του εκπαιδευτικού υλικού όσο και τη δέσμευση των χρηστών. Υπολογίζεται με τον τύπο:

Completion Rate (%) = (Αριθμός χρηστών που ολοκλήρωσαν / Αριθμός χρηστών που ξεκίνησαν) × 100

Η παρακολούθηση γίνεται μέσω συστήματος διαχείρισης μαθημάτων (LMS) ή μέσω custom tracking events (π.χ. “completed lesson” flags). Παράγοντες που επηρεάζουν το ποσοστό περιλαμβάνουν τη διάρκεια του course, τη διαδραστικότητα, την προσβασιμότητα και την ποιότητα των quiz ή ασκήσεων. Ένας υψηλός δείκτης θεωρείται ένδειξη εκπαιδευτικής αποτελεσματικότητας και ικανοποίησης χρηστών.

22. Average Revenue per User (ARPU)

Το *Average Revenue per User (ARPU)* μετρά το μέσο έσοδο που παράγεται ανά ενεργό χρήστη σε μια συγκεκριμένη χρονική περίοδο (συνήθως μηνιαία ή τριμηνιαία). Είναι βασικός δείκτης για την αξιολόγηση της οικονομικής απόδοσης μιας πλατφόρμας ή ψηφιακής υπηρεσίας, ιδίως σε μοντέλα συνδρομής ή freemium. Υπολογίζεται με τον τύπο:

ARPU = Συνολικά έσοδα / Αριθμός ενεργών χρηστών

Για την ακρίβεια του υπολογισμού, ως "ενεργοί χρήστες" λαμβάνονται συνήθως οι MAU (Monthly Active Users), ενώ στα έσοδα περιλαμβάνονται όλες οι σχετικές πηγές (συνδρομές, αγορές εντός εφαρμογής, διαφημιστικά έσοδα κ.λπ.). Ο δείκτης βοηθά στη σύγκριση περιόδων, την εκτίμηση της αποδοτικότητας στρατηγικών τιμολόγησης και την πρόβλεψη εσόδων με βάση την ανάπτυξη χρήσης.

23. Conversion Rate (Free to Paid Users)

Ο δείκτης Conversion Rate (Free to Paid Users) μετρά το ποσοστό των χρηστών που μετατρέπονται από δωρεάν χρήση της πλατφόρμας σε πληρωμένη. Ο υπολογισμός του βασίζεται στον τύπο:

CR=(Νέοι επί πληρωμή χρήστες / Δωρεάν ενεργοί χρήστες για το ίδιο διάστημα) × 100.

Τα απαραίτητα δεδομένα προκύπτουν είτε από το σύστημα διαχείρισης χρηστών (CRM, database), είτε μέσω εργαλείων ανάλυσης funnel conversion (π.χ. Google Analytics, Mixpanel). Ο δείκτης είναι κρίσιμος για επιχειρηματικά μοντέλα freemium ή SaaS, καθώς δείχνει την ικανότητα του προϊόντος να μετατρέπει την επισκεψιμότητα σε έσοδα.

24. Customer Acquisition Cost (CAC)

Το Customer Acquisition Cost (CAC) εκφράζει το συνολικό μέσο κόστος που απαιτείται για την απόκτηση ενός νέου πελάτη. Περιλαμβάνει όλα τα σχετικά έξοδα marketing, διαφημίσεων, καμπανιών, μισθοδοσίας τμημάτων πωλήσεων ή εξωτερικών συνεργατών. Ο δείκτης υπολογίζεται με τον τύπο: CAC = (Συνολικές δαπάνες απόκτησης πελατών / Αριθμός νέων πελατών σε δεδομένο διάστημα). Χρησιμοποιείται ευρέως για την αξιολόγηση της βιωσιμότητας μιας επιχείρησης και τη βελτιστοποίηση της στρατηγικής προσέλκυσης.

25. Click-Through Rate (CTR)

Ο Click-Through Rate (CTR) μετρά την αποτελεσματικότητα μιας διαφημιστικής καμπάνιας ή στοιχείου (π.χ. banner, email, CTA) ως προς την ικανότητά του να παρακινήσει τον χρήστη να αλληλεπιδράσει. Υπολογίζεται ως:

CTR = (Αριθμός κλικ / Αριθμός εμφανίσεων) × 100.

Όσο υψηλότερος είναι ο δείκτης, τόσο μεγαλύτερη η σχετικότητα και αποδοτικότητα του περιεχομένου για το κοινό στόχευσης. Ο CTR παρακολουθείται μέσω πλατφορμών διαφημίσεων (Google Ads, Facebook Ads) και χρησιμοποιείται για τη βελτιστοποίηση του περιεχομένου και της στόχευσης.

26. Deployment Frequency

Ο δείκτης Deployment Frequency αποτυπώνει πόσο συχνά πραγματοποιούνται αναβαθμίσεις, νέες εκδόσεις ή λειτουργικές αλλαγές σε περιβάλλον παραγωγής. Η συχνότητα υπολογίζεται ως πλήθος deploys ανά ημέρα, εβδομάδα ή μήνα. Παρακολουθείται συνήθως μέσω CI/CD εργαλείων όπως GitLab CI, GitHub Actions ή Jenkins. Υψηλή τιμή δείχνει ώριμη DevOps πρακτική και συνεχόμενη παροχή αξίας στον τελικό χρήστη, ενώ χαμηλή μπορεί να δηλώνει περιορισμούς στην εσωτερική ροή ή τεχνικά χρέη.

27. Bug Fix Time

Το Bug Fix Time μετρά τον μέσο χρόνο που απαιτείται από τη στιγμή αναφοράς ενός σφάλματος (bug) μέχρι την πλήρη επίλυσή του και την εγκατάστασή του σε παραγωγή. Υπολογίζεται με βάση τα δεδομένα που καταγράφονται στα συστήματα υποστήριξης και παρακολούθησης

εργασιών (π.χ. Jira, GitHub Issues). Ο δείκτης εκφράζεται σε ώρες ή ημέρες και αποτυπώνει την ικανότητα της ομάδας να ανταποκρίνεται σε τεχνικά προβλήματα με αποτελεσματικότητα και ταχύτητα.

28. System Monitoring & Logging

Αυτός ο δείκτης αφορά το επίπεδο πληρότητας και αξιοπιστίας των μηχανισμών παρακολούθησης και καταγραφής γεγονότων στο σύστημα. Περιλαμβάνει την εγκατάσταση και ρύθμιση εργαλείων όπως Prometheus, Grafana, ELK stack (Elasticsearch, Logstash, Kibana) για real-time monitoring και centralized logging. Η αξιολόγηση γίνεται είτε ποιοτικά (αν υπάρχουν alerts, dashboards, coverage), είτε ποσοτικά (π.χ. ποσοστό επιτυχών ανιχνεύσεων προβλημάτων πριν αναφερθούν από χρήστες).

29. Development Time

Ο δείκτης Development Time καταγράφει τον χρόνο που απαιτείται για την υλοποίηση ενός νέου χαρακτηριστικού ή υποσυστήματος, από τη στιγμή που θα ανατεθεί μέχρι να παραδοθεί προς δοκιμή ή παραγωγή. Η μέτρηση γίνεται με βάση τα timestamps σε εργαλεία project management (π.χ. Jira, Trello) ή Git commits. Το KPI χρησιμοποιείται για την εκτίμηση του ρυθμού παραγωγής, τον υπολογισμό κόστους ανά λειτουργία και τη βελτιστοποίηση των workflow ανάπτυξης.

30. Ease of Integration

Το Ease of Integration μετρά τον βαθμό δυσκολίας που παρουσιάζει η ενσωμάτωση τρίτων υπηρεσιών ή APIs (π.χ. Stripe, Firebase, Auth0) στην υπάρχουσα εφαρμογή. Η αξιολόγηση βασίζεται στο πλήθος των απαιτούμενων βημάτων, τον χρόνο υλοποίησης, την ποιότητα τεκμηρίωσης των APIs και την ανάγκη για debugging. Ένα χαμηλό επίπεδο δυσκολίας συνεπάγεται ευελιξία και ταχύτερη κυκλοφορία νέων δυνατοτήτων.

31. Developer Learning Curve

Ο δείκτης αυτός αποτυπώνει το χρονικό διάστημα που χρειάζεται ένας νέος προγραμματιστής ώστε να εξοικειωθεί με το περιβάλλον ανάπτυξης, την αρχιτεκτονική και τα εργαλεία του έργου. Υπολογίζεται μέσω ποιοτικής αξιολόγησης (π.χ. συνεντεύξεις onboarding) ή μέσω του χρόνου που απαιτείται για να αναλάβει επιτυχώς ένα first task. Μικρότερη καμπύλη εκμάθησης σημαίνει ταχύτερη παραγωγικότητα και μειωμένο κόστος ένταξης νέου προσωπικού.

32. Code Reusability

Το Code Reusability αφορά το ποσοστό του πηγαίου κώδικα που μπορεί να χρησιμοποιηθεί αυτούσιο ή με ελάχιστες αλλαγές σε άλλα έργα ή λειτουργίες. Ο δείκτης εκτιμάται αναλύοντας

την αρχιτεκτονική του λογισμικού (modular design, reusable components, separation of concerns) και υποστηρίζεται μέσω συστημάτων versioning (π.χ. npm packages, shared libraries). Η επαναχρησιμοποίηση μειώνει τον χρόνο ανάπτυξης, περιορίζει τα σφάλματα και αυξάνει την ομοιογένεια του λογισμικού.

33. Documentation Quality

Ο δείκτης Documentation Quality αξιολογεί την πληρότητα, σαφήνεια, δομή και ενημερότητα της τεχνικής τεκμηρίωσης. Η τεκμηρίωση μπορεί να περιλαμβάνει API references, οδηγούς εγκατάστασης, αρχιτεκτονικά διαγράμματα και οδηγίες troubleshooting. Η αξιολόγηση γίνεται μέσω peer review, auto-generated coverage tools (π.χ. DocFx, JSDoc), και ανατροφοδότησης από την ομάδα ανάπτυξης. Η καλή τεκμηρίωση είναι κρίσιμη για τη μακροχρόνια υποστήριξη και συντήρηση του συστήματος.

34. Ease of Deployment

Ο δείκτης Ease of Deployment αξιολογεί τον βαθμό δυσκολίας και πολυπλοκότητας της διαδικασίας εγκατάστασης μιας νέας έκδοσης του λογισμικού στο περιβάλλον παραγωγής. Περιλαμβάνει παραμέτρους όπως η ανάγκη για downtime, η αυτοματοποίηση μέσω εργαλείων CI/CD (π.χ. GitHub Actions, Jenkins), η απαίτηση για χειροκίνητες παρεμβάσεις και η δυνατότητα προεπισκόπησης. Η μέτρηση μπορεί να βασιστεί στον χρόνο ολοκλήρωσης της διαδικασίας ανά έκδοση και στον αριθμό των απαιτούμενων βημάτων.

35. Roll-back Time

Το Roll-back Time αφορά τον χρόνο που απαιτείται για να επανέλθει το σύστημα σε προηγούμενη, σταθερή κατάσταση σε περίπτωση αποτυχίας της νέας έκδοσης. Ο υπολογισμός βασίζεται στην ύπαρξη αυτοματοποιημένων μηχανισμών επαναφοράς, snapshots, και version control των εξαρτημάτων του συστήματος. Ο δείκτης είναι κρίσιμος για τη διασφάλιση διαθεσιμότητας και αξιοπιστίας της υπηρεσίας.

36. Extensibility

Ο δείκτης Extensibility εκτιμά την ευκολία με την οποία μπορούν να προστεθούν νέες λειτουργίες ή να επεκταθεί η υπάρχουσα υποδομή της εφαρμογής. Περιλαμβάνει στοιχεία όπως modular design, χρήση plugin architecture ή APIs, και υποστήριξη για integration patterns. Η αξιολόγηση μπορεί να προκύψει εμπειρικά μέσα από παραδείγματα επεκτάσεων και τον χρόνο που αυτές απαιτούν.

37. Code Complexity

Ο δείκτης Code Complexity μετρά την πολυπλοκότητα του πηγαίου κώδικα, συνήθως μέσω εργαλείων στατιστικής ανάλυσης όπως SonarQube, Code Climate ή μέτρων όπως η Cyclomatic

Complexity. Πολύπλοκος κώδικας σημαίνει αυξημένο τεχνικό χρέος και δυσκολία συντήρησης. Ο στόχος είναι να διατηρείται η πολυπλοκότητα σε αποδεκτά επίπεδα ώστε η ανάπτυξη να είναι εύκολη και ευέλικτη.

38. Time to Market

Ο δείκτης Time to Market αναφέρεται στον συνολικό χρόνο που απαιτείται για την κυκλοφορία ενός νέου χαρακτηριστικού ή προϊόντος, από την αρχική ιδέα μέχρι την υλοποίηση σε παραγωγή. Η μέτρηση ξεκινά από την καταγραφή ενός request στο backlog και ολοκληρώνεται με το release. Είναι κρίσιμος για την ανταγωνιστικότητα και την ικανότητα γρήγορης ανταπόκρισης στις απαιτήσεις της αγοράς.

39. Change Adoption Rate

Ο Change Adoption Rate μετρά το ποσοστό των χρηστών ή εσωτερικών ομάδων που υιοθετούν νέες λειτουργίες ή διαδικασίες. Ο δείκτης βασίζεται σε δεδομένα χρήσης χαρακτηριστικών, feedback ή εσωτερικές έρευνες. Υψηλό adoption υποδηλώνει ότι η αλλαγή ήταν επιτυχημένη, χρήσιμη και κατανοητή από το κοινό.

40. Feature Deployment Speed

Ο Feature Deployment Speed εκφράζει τον χρόνο που απαιτείται από τη στιγμή που μια λειτουργία ορίζεται ως απαίτηση μέχρι την ανάπτυξή της σε παραγωγή. Περιλαμβάνει τη φάση ανάλυσης, ανάπτυξης, ελέγχου και τελικής ένταξης. Ο δείκτης είναι κρίσιμος για agile περιβάλλοντα και ομάδες που εφαρμόζουν continuous delivery πρακτικές.

41. Adaptation to Market Trends

Αυτός ο δείκτης αποτυπώνει την ικανότητα της επιχείρησης ή της πλατφόρμας να προσαρμόζεται σε τεχνολογικές, καταναλωτικές ή νομοθετικές τάσεις της αγοράς. Η μέτρηση είναι περισσότερο ποιοτική, αν και μπορεί να εκτιμηθεί με βάση τον αριθμό ή τον ρυθμό υλοποίησης σχετικών πρωτοβουλιών ανά έτος.

42. Decision-Making Speed

Ο δείκτης Decision-Making Speed μετρά τον χρόνο που απαιτείται για την έγκριση ή λήψη κρίσιμων αποφάσεων, είτε σε επίπεδο προϊόντος, είτε σε τεχνολογική κατεύθυνση. Επηρεάζει άμεσα την ταχύτητα εκτέλεσης έργων και τη γενικότερη ευελιξία του οργανισμού.

43. Cross-Functional Collaboration Efficiency

Αποτυπώνει την αποτελεσματικότητα συνεργασίας μεταξύ ομάδων με διαφορετικά αντικείμενα (π.χ. ανάπτυξης, σχεδιασμού, μάρκετινγκ). Ο δείκτης μπορεί να αξιολογηθεί μέσα από τον αριθμό καθυστερήσεων που οφείλονται σε ασυνεννοησία, την ταχύτητα feedback loop ή μέσα από employee satisfaction surveys.

44. Process Automation Rate

Ο Process Automation Rate δείχνει το ποσοστό των επιχειρησιακών διαδικασιών που έχουν αυτοματοποιηθεί, όπως το deployment, testing, billing ή onboarding. Υπολογίζεται ως

PAR=(Αυτοματοποιημένες διαδικασίες / Σύνολο βασικών διαδικασιών) × 100.

Η αυτοματοποίηση αυξάνει την αποδοτικότητα και μειώνει τα ανθρώπινα λάθη.

45. Innovation Rate

Ο δείκτης Innovation Rate μετρά τον αριθμό νέων χαρακτηριστικών, τεχνολογιών ή προϊόντων που υλοποιήθηκαν σε ένα συγκεκριμένο διάστημα (π.χ. ανά τρίμηνο ή έτος). Μπορεί να προκύψει από το backlog ή από τις αναφορές R&D. Αποτυπώνει τον ρυθμό καινοτομίας του οργανισμού.

46. Regulatory Compliance Adaptability

Αυτός ο δείκτης αξιολογεί την ταχύτητα και την ικανότητα προσαρμογής στις νέες κανονιστικές απαιτήσεις (π.χ. GDPR, WCAG, ISO). Υπολογίζεται με βάση τον χρόνο συμμόρφωσης από την ημερομηνία ανακοίνωσης της απαίτησης μέχρι την πλήρη εφαρμογή των μέτρων συμμόρφωσης.

47. User Growth Handling

Ο δείκτης User Growth Handling μετρά την ικανότητα της υποδομής και του λογισμικού να ανταποκριθούν στην ταχεία αύξηση του αριθμού χρηστών. Η αξιολόγηση γίνεται μέσω stress testing, benchmarking και παρακολούθησης της σταθερότητας του συστήματος σε σενάρια scaling.

48. Latency under Load

Το Latency under Load αποτυπώνει τον μέσο χρόνο απόκρισης του συστήματος κατά τη διάρκεια αιχμής ή φορτίου, όταν εξυπηρετεί μεγάλο αριθμό ταυτόχρονων αιτήσεων. Η μέτρηση γίνεται με εργαλεία όπως k6, JMeter ή Locust.

49. Infrastructure Elasticity

Ο δείκτης αυτός αναφέρεται στην ικανότητα της υποδομής (cloud, servers) να αυξάνει ή να μειώνει τους πόρους της (CPU, memory, instances) αυτόματα ανάλογα με τις ανάγκες της εφαρμογής. Εργαλεία όπως AWS Auto Scaling ή Kubernetes HPA χρησιμοποιούνται για αξιολόγηση.

50. Horizontal vs Vertical Scaling Efficiency

Αξιολογεί τη σχετική αποτελεσματικότητα της οριζόντιας (προσθήκη servers) έναντι της κάθετης (ενίσχυση εξοπλισμού) κλιμάκωσης. Ο δείκτης βασίζεται σε μετρήσεις κόστους, επίδοσης και διαθεσιμότητας για κάθε μέθοδο.

51. Database Scalability

Ο δείκτης αυτός μετρά την ικανότητα της βάσης δεδομένων να εξυπηρετεί αυξανόμενο όγκο δεδομένων και αιτήσεων χωρίς σημαντική υποβάθμιση επιδόσεων. Περιλαμβάνει κριτήρια όπως read/write throughput, indexing, sharding και replication.

52. Cost per Additional 1,000 Users

Αυτός ο δείκτης υπολογίζει το πρόσθετο λειτουργικό κόστος για την υποστήριξη κάθε 1.000 νέων χρηστών. Περιλαμβάνει έξοδα υποδομής (servers, storage), υποστήριξης (support tickets) και αδειοδοτήσεων (licenses).

53. Server Utilization Rate

Ο Server Utilization Rate μετρά την πραγματική χρήση των διαθέσιμων πόρων των servers (CPU, RAM, IO) σε περιόδους αιχμής και ηρεμίας. Η μέτρηση γίνεται μέσω εργαλείων monitoring και βοηθά στην εξοικονόμηση κόστους μέσω βελτιστοποίησης της υποδομής.

54. Multi-Region Deployment Readiness

Ο δείκτης αυτός εξετάζει την ετοιμότητα της πλατφόρμας να αναπτυχθεί σε πολλαπλές γεωγραφικές περιοχές, ώστε να διασφαλίζεται χαμηλή καθυστέρηση, ανθεκτικότητα και συμμόρφωση με τοπικές απαιτήσεις. Περιλαμβάνει αξιολόγηση data replication, failover μηχανισμών και CDN υποδομής.

55. API Request Throughput

Μετρά τον μέγιστο αριθμό αιτήσεων (requests) που μπορεί να διαχειριστεί το API ανά δευτερόλεπτο υπό κανονική λειτουργία. Η αξιολόγηση βασίζεται σε benchmarks και φορτισμένες δοκιμές (stress tests), και δείχνει την επεκτασιμότητα της εφαρμογής σε επίπεδο backend.

56. Performance Degradation Rate

Ο δείκτης αυτός αποτυπώνει τον ρυθμό υποβάθμισης των επιδόσεων του συστήματος καθώς αυξάνεται ο φόρτος ή τα δεδομένα. Υπολογίζεται παρατηρώντας πώς μεταβάλλεται η ταχύτητα απόκρισης, ο ρυθμός σφαλμάτων ή η χρήση πόρων κατά την αύξηση του όγκου χρηστών και συναλλαγών.

5.3.6. Κριτήρια αξιολόγησης

Για τη συγκριτική αξιολόγηση των δύο τεχνολογικών υλοποιήσεων, καθορίστηκαν δέκα βασικές **κατηγορίες κριτηρίων**, οι οποίες καλύπτουν όλες τις κρίσιμες πτυχές ενός σύγχρονου, λειτουργικού και παραγωγικού πληροφοριακού συστήματος. Παρακάτω παρουσιάζονται αναλυτικά:

1. Απόδοση (Performance)

Αξιολογείται η ταχύτητα και η σταθερότητα του συστήματος κατά την εκτέλεση βασικών λειτουργιών. Περιλαμβάνει δείκτες όπως ο χρόνος απόκρισης, ο χρόνος φόρτωσης σελίδων και η ικανότητα διαχείρισης ταυτόχρονων χρηστών.

2. Διαθεσιμότητα (Availability)

Αναφέρεται στην αξιοπιστία και την απρόσκοπτη λειτουργία της πλατφόρμας. Εξετάζεται το ποσοστό uptime, ο χρόνος αποκατάστασης σε περίπτωση σφάλματος και η γενικότερη ανθεκτικότητα του συστήματος.

3. Ασφάλεια (Security)

Αφορά την προστασία δεδομένων και των χρηστών. Περιλαμβάνει την ύπαρξη κρυπτογράφησης, την ανθεκτικότητα σε επιθέσεις και τη διαχείριση ρόλων και προσβάσεων.

4. Κόστος (Cost)

Συνυπολογίζεται το συνολικό κόστος υλοποίησης και λειτουργίας: από την αρχική ανάπτυξη έως τη φιλοξενία, τη συντήρηση, τις άδειες χρήσης και την εξάρτηση από τρίτους.

5. Χρήστες (User Metrics)

Μετρώνται δείκτες που αφορούν τη συμπεριφορά και την εμπειρία των τελικών χρηστών, όπως ο βαθμός ικανοποίησης, η διάρκεια παραμονής, η συμμετοχή και η επιτυχής ολοκλήρωση μαθημάτων.

6. Marketing

Αξιολογούνται οικονομικά KPIs που σχετίζονται με τα έσοδα, την απόδοση διαφημίσεων, την απόκτηση πελατών και τη μετατροπή χρηστών από δωρεάν σε πληρωμένους.

7. Software Maintenance

Περιλαμβάνει την ευκολία στη διαχείριση του κώδικα και των αναβαθμίσεων, την ταχύτητα επιδιόρθωσης σφαλμάτων, την ύπαρξη εργαλείων παρακολούθησης και τις δυνατότητες rollback.

8. Development Ease

Αναφέρεται στη δυσκολία ανάπτυξης του συστήματος: χρόνος εκμάθησης, ενσωμάτωση APIs, τεκμηρίωση, επαναχρησιμοποίηση κώδικα και γενική εμπειρία προγραμματιστή.

9. Business Agility

Μετρά την ταχύτητα με την οποία η τεχνολογία επιτρέπει στην επιχείρηση να προσαρμόζεται σε αλλαγές, να κυκλοφορεί νέες λειτουργίες και να αυτοματοποιεί διαδικασίες.

10. Scalability (Κλιμάκωση)

Αξιολογείται η ικανότητα της πλατφόρμας να διαχειρίζεται αύξηση φορτίου, χρηστών και δεδομένων χωρίς να μειώνεται η απόδοση.

5.3.7. Συγκριτικός πίνακας και ανάλυση αποτελεσμάτων

Στον παρακάτω πίνακα παρουσιάζονται τα βασικά KPI (Key Performance Indicators) που χρησιμοποιήθηκαν για την αξιολόγηση των δύο τεχνολογικών επιλογών: της λύσης με WordPress και της λύσης με Microservices. Η βαθμολόγηση κάθε παραμέτρου έγινε αποκλειστικά με βάση προσωπική εμπειρία και εμπειρική παρατήρηση κατά τη διάρκεια της υλοποίησης, χωρίς να χρησιμοποιηθούν δημοσιευμένα δεδομένα ή μέσοι όροι από τη βιβλιογραφία.

Στόχος της συγκριτικής αξιολόγησης είναι η αποτύπωση των λειτουργικών διαφορών και η αναγνώριση των ισχυρών και αδύναμων σημείων κάθε προσέγγισης, όπως αυτά αναδείχθηκαν μέσα από την πράξη και όχι μέσα από θεωρητικά μοντέλα.

Για λόγους συνέπειας και σαφήνειας, εφαρμόστηκε κοινή βαθμολογική κλίμακα από -5 έως +5, όπου το +5 αντιστοιχεί στην καλύτερη δυνατή επίδοση, το 0 σε ουδέτερη/μέτρια επίδοση, ενώ το -5 σε πολύ κακή

επίδοση. Η κλίμακα αυτή εξομαλύνει την αποτίμηση τόσο για δείκτες με θετική φορά (π.χ. ρυθμός απόδοσης, ευκολία επέκτασης), όσο και για δείκτες με αρνητική φορά (π.χ. κόστος, χρόνος απόκρισης, downtime), καθιστώντας τις επιδόσεις απευθείας συγκρίσιμες.

Κατηγορία	KPI	Περιγραφή	WordPress Score	Microservices Score
Απόδοση	Page Load Time	Χρόνος φόρτωσης σελίδας	2	4
Απόδοση	Server Response Time	Χρόνος απόκρισης server	1	4
Απόδοση	Time to First Byte (TTFB)	Πόσο γρήγορα απαντά ο server στο πρώτο request	1	4
Απόδοση	Time to Interactive (TTI)	Πότε η σελίδα είναι πλήρως διαδραστική	2	4
Απόδοση	Concurrent Users Handling	Ταυτόχρονοι χρήστες πριν επηρεαστεί η απόδοση	0	4
Διαθεσιμότητα	Uptime & Availability	Ποσοστό διαθεσιμότητας	2	4
Διαθεσιμότητα	Error Rate	Ποσοστό αποτυχημένων requests	1	4
Διαθεσιμότητα	Downtime per Month	Χρόνος μη διαθεσιμότητας	1	4
Διαθεσιμότητα	Backup & Recovery Time	Χρόνος αποκατάστασης συστήματος	1	3
Ασφάλεια	Vulnerabilities Detected	Αριθμός ανιχνευμένων ευπαθειών	0	4
Ασφάλεια	Number of Security Incidents	Αριθμός περιστατικών ασφαλείας	1	4
Ασφάλεια	Encryption Level	Επίπεδο κρυπτογράφησης δεδομένων	1	4
Ασφάλεια	Data Breach Risk	Πιθανότητα διαρροής δεδομένων	0	4
Κόστος	Initial Development Cost	Κόστος αρχικής ανάπτυξης	4	-3
Κόστος	Monthly Infrastructure Cost	Κόστος hosting, cloud, servers	2	1
Κόστος	Maintenance & Support Cost	Κόστος συντήρησης & υποστήριξης	1	3
Κόστος	3rd Party Service Costs	Κόστος APIs, licenses, plugins	0	4
Χρήστες	User Churn Rate	Ποσοστό χρηστών που εγκαταλείπουν την πλατφόρμα	1	3
Χρήστες	Active Users per Month (MAU)	Πόσοι χρήστες είναι ενεργοί μηνιαίως	1	3
Χρήστες	Customer Satisfaction Score (CSAT)	Μέτρηση ικανοποίησης χρηστών	2	3
Χρήστες	Course Completion Rate	Ποσοστό χρηστών που ολοκληρώνουν courses	1	3

Marketing	Average Revenue per User (ARPU)	Μέσο έσοδο ανά χρήστη	1	3
Marketing	Conversion Rate (Free to Paid Users)	Πόσοι χρήστες μετατρέπονται σε πληρωμένους	0	2
Marketing	Customer Acquisition Cost (CAC)	Κόστος απόκτησης πελάτη	2	1
Marketing	Click-Through Rate (CTR)	Αποδοτικότητα διαφημίσεων	1	2
Software Maintenance	Deployment Frequency	Συχνότητα αναβαθμίσεων	0	4
Software Maintenance	Bug Fix Time	Μέσος χρόνος επίλυσης bugs	1	4
Software Maintenance	System Monitoring & Logging	Επίπεδο παρακολούθησης και logging για troubleshooting	-1	4
Development Ease	Development Time	Χρόνος που απαιτείται για την αρχική ανάπτυξη	4	0
Development Ease	Ease of Integration	Πόσο εύκολη είναι η ενσωμάτωση τρίτων υπηρεσιών ή APIs	1	3
Development Ease	Developer Learning Curve	Χρόνος που χρειάζονται οι προγραμματιστές για να μάθουν την τεχνολογία	4	0
Development Ease	Code Reusability	Πόσος κώδικας μπορεί να επαναχρησιμοποιηθεί για μελλοντικά projects	0	4
Development Ease	Documentation Quality	Ποιότητα τεκμηρίωσης για ανάπτυξη & υποστήριξη	1	3
Software Maintenance	Ease of Deployment	Ευκολία εγκατάστασης updates	3	4
Software Maintenance	Roll-back Time	Χρόνος επαναφοράς σε προηγούμενη έκδοση	1	3
Software Maintenance	Extensibility	Δυνατότητα προσθήκης νέων λειτουργιών	1	4
Software Maintenance	Code Complexity	Πολυπλοκότητα κώδικα	-1	3
Business Agility	Time to Market	Χρόνος ανάπτυξης και κυκλοφορίας νέων προϊόντων/υπηρεσιών.	4	1
Business Agility	Change Adoption Rate	Ποσοστό επιτυχούς υιοθέτησης νέων διαδικασιών ή τεχνολογιών.	1	4
Business Agility	Feature Deployment Speed	Ταχύτητα με την οποία αναπτύσσονται και εφαρμόζονται νέες λειτουργίες.	1	4
Business Agility	Adaptation to Market Trends	Ταχύτητα και ικανότητα προσαρμογής σε αλλαγές της αγοράς.	0	4

Business Agility	Decision-Making Speed	Ταχύτητα λήψης επιχειρηματικών αποφάσεων.	1	3
Business Agility	Cross-Functional Collaboration Efficiency	Αποδοτικότητα συνεργασίας μεταξύ διαφορετικών ομάδων.	1	3
Business Agility	Process Automation Rate	Ποσοστό επιχειρηματικών διαδικασιών που έχουν αυτοματοποιηθεί.	0	4
Business Agility	Innovation Rate	Αριθμός νέων πρωτοβουλιών, υπηρεσιών ή προϊόντων ανά έτος.	0	4
Business Agility	Regulatory Compliance Adaptability	Ταχύτητα συμμόρφωσης με νέους κανονισμούς και νομοθεσίες.	0	4
Scalability	User Growth Handling	Μέγιστος αριθμός χρηστών που μπορεί να διαχειριστεί το σύστημα χωρίς υποβάθμιση της απόδοσης.	0	4
Scalability	Latency under Load	Χρόνος απόκρισης του συστήματος υπό αυξημένο φορτίο.	0	4
Scalability	Infrastructure Elasticity	Δυνατότητα αύξησης ή μείωσης πόρων (servers, databases) ανάλογα με τη ζήτηση.	-1	4
Scalability	Horizontal vs Vertical Scaling Efficiency	Αποδοτικότητα της κλιμάκωσης σε περισσότερους servers (horizontal) ή καλύτερο hardware (vertical).	-1	4
Scalability	Database Scalability	Πόσο καλά μπορεί να επεκταθεί η βάση δεδομένων χωρίς μείωση της απόδοσης.	0	4
Scalability	Cost per Additional 1,000 Users	Κόστος υποστήριξης κάθε 1.000 νέων χρηστών.	1	3
Scalability	Server Utilization Rate	Αξιοποίηση των servers σε περιόδους υψηλής και χαμηλής χρήσης.	0	4
Scalability	Multi-Region Deployment Readiness	Ικανότητα ανάπτυξης της εφαρμογής σε πολλαπλά data centers ή cloud regions.	-1	4
Scalability	API Request Throughput	Μέγιστος αριθμός API requests που μπορεί να διαχειριστεί το σύστημα ανά δευτερόλεπτο.	0	4
Scalability	Performance Degradation Rate	Ρυθμός μείωσης της απόδοσης καθώς αυξάνεται ο όγκος των δεδομένων και των χρηστών.	-1	4
		Total Score	48	182

Πίνακας 5.1: Δείκτες Key Performance Indicators σύγκρισης των δυο αρχιτεκτονικών

Η προσέγγιση που ακολουθήθηκε δεν περιορίστηκε μόνο στην τεχνική παρατήρηση ή τη θεωρητική προσέγγιση, αλλά στηρίχθηκε και στην εμπειρία από την ίδια την υλοποίηση των δύο λύσεων, με σκοπό την όσο το δυνατόν πιο ρεαλιστική αποτύπωση των πλεονεκτημάτων και των περιορισμών κάθε τεχνολογίας.

Συνολικά, η πλατφόρμα WordPress παρουσιάζει υψηλή ταχύτητα υλοποίησης και σημαντικά χαμηλότερο αρχικό κόστος, προσφέροντας μια άμεση λύση για δημιουργία εκπαιδευτικών portals. Ειδικά όταν οι απαιτήσεις είναι περιορισμένες, η έτοιμη υποδομή που προσφέρει το CMS και τα LMS plugins, δίνει τη δυνατότητα γρήγορης παραγωγής ενός MVP. Παρ' όλα αυτά, η ίδια ακριβώς αρχιτεκτονική που ευνοεί την απλότητα, περιορίζει σημαντικά τις δυνατότητες παραμετροποίησης, επέκτασης και απόδοσης σε συνθήκες πίεσης ή αύξησης κλίμακας.

Αντίθετα, η λύση βασισμένη σε αρχιτεκτονική microservices καταγράφει σταθερά υψηλότερες επιδόσεις στους περισσότερους κρίσιμους δείκτες, ιδίως σε θέματα διαθεσιμότητας, αποδοτικότητας, επεκτασιμότητας και ανθεκτικότητας. Η δυνατότητα διαχωρισμού των λειτουργιών σε ανεξάρτητες υπηρεσίες, η χρήση containerization (π.χ. Docker), η ενσωμάτωση εργαλείων όπως API gateways, queue systems, monitoring tools και η ύπαρξη CI/CD pipelines, οδηγούν σε ένα τεχνικά ώριμο, ευέλικτο και κλιμακούμενο σύστημα. Το κύριο μειονέκτημα είναι ο αυξημένος χρόνος ανάπτυξης και η τεχνική πολυπλοκότητα, που απαιτεί έμπειρη ομάδα και δομημένες διαδικασίες DevOps.

Αξιολογώντας τους επιμέρους δείκτες:

- Σε KPIs όπως Page Load Time, Server Response Time, Concurrent Users Handling και API Throughput, τα microservices υπερτερούν καθαρά.
- Σε δείκτες κόστους όπως Initial Development Cost και Learning Curve, το WordPress έχει σαφές πλεονέκτημα.
- Όσον αφορά την Επιχειρησιακή Ευελιξία και την Κλιμάκωση, η λύση με microservices εμφανίζει συντριπτικά καλύτερα αποτελέσματα, καθιστώντας την ιδανική επιλογή για μακροπρόθεσμα πλάνα ανάπτυξης.

Καταλήγοντας, η επιλογή μεταξύ των δύο λύσεων δεν είναι απόλυτη. Το WordPress είναι ιδανικό για MVPs, γρήγορη παρουσίαση concept ή έργα με χαμηλό προϋπολογισμό και τεχνική υποστήριξη. Η αρχιτεκτονική microservices ενδείκνυται για περιβάλλοντα όπου απαιτούνται ελεγχόμενη απόδοση, επεκτασιμότητα, ευελιξία και τεχνική βιωσιμότητα σε βάθος χρόνου. Η τελική επιλογή εξαρτάται από τις στρατηγικές ανάγκες του έργου, το προσδοκώμενο μέγεθος και το διαθέσιμο ανθρώπινο και τεχνολογικό κεφάλαιο.

6. Συμπεράσματα και Μελλοντική Εργασία

6.1. Επισκόπηση αποτελεσμάτων και μεθοδολογίας

Στην παρούσα διπλωματική εργασία εφαρμόστηκε η μεθοδολογία **System Analysis & Design** για την συνολική προσέγγιση στην ανάπτυξη μίας πλατφόρμας online μαθημάτων, με στόχο τη σύγκριση δύο διαφορετικών τεχνολογικών προσεγγίσεων: **WordPress** και **Microservices**.

Αρχικά, έγινε καταγραφή των απαιτήσεων του συστήματος και σχεδιάστηκε ένα γενικό λογικό μοντέλο λειτουργίας (DFD Level 0 και 1). Ορίστηκε ένα Minimum Viable Product (MVP), το οποίο αποτέλεσε τον άξονα των υλοποιήσεων. Στη συνέχεια, κατασκευάστηκαν δύο λειτουργικές πλατφόρμες. Η πρώτη βασισμένη σε CMS (WordPress με Tutor LMS), και η δεύτερη υλοποιημένη με χρήση ανεξάρτητων υπηρεσιών (Microservices αρχιτεκτονική).

Για την αξιολόγηση, χρησιμοποιήθηκε ένα ολοκληρωμένο σύνολο **KPIs** οργανωμένο σε δέκα κατηγορίες. Η συγκριτική ανάλυση βάσει αυτών των δεικτών κατέδειξε τα πλεονεκτήματα και τις αδυναμίες κάθε προσέγγισης. Συγκεκριμένα, η λύση με WordPress υπερέχει σε ευκολία και ταχύτητα ανάπτυξης, ενώ η προσέγγιση με microservices διακρίνεται σε απόδοση, ευελιξία και επεκτασιμότητα.

6.2. Τεχνικά και επιχειρησιακά οφέλη από την εργασία

Η υλοποίηση της εργασίας προσέφερε σημαντική τεχνική εμπειρία τόσο σε επίπεδο frontend/backend ανάπτυξης, όσο και σωστή διαχείριση υπηρεσιών. Επιπλέον, επιτεύχθηκε ουσιαστική εξοικείωση με εργαλεία και τεχνολογίες όπως WordPress LMS plugins, Django, REST APIs.

Σε επιχειρησιακό επίπεδο, αναπτύχθηκαν δεξιότητες που αφορούν στην ανάλυση απαιτήσεων, στη λήψη αποφάσεων βάσει δεδομένων, καθώς και στην αξιολόγηση τεχνολογικών επενδύσεων με μετρήσιμα κριτήρια (KPIs). Η εφαρμογή συγκεντρώνει λειτουργικά χαρακτηριστικά που συναντώνται σε σύγχρονες εκπαιδευτικές πλατφόρμες, ενώ παράλληλα δίνει προοπτικές για επιχειρηματική αξιοποίηση.

6.3. Προτάσεις για μελλοντική βελτίωση ή επέκταση του συστήματος

Στο πλαίσιο μελλοντικής εξέλιξης, προτείνονται οι εξής κατευθύνσεις:

- Υλοποίηση Hybrid αρχιτεκτονικής, συνδυάζοντας την ταχύτητα υλοποίησης του WordPress με επιλεγμένες microservices για κρίσιμες λειτουργίες.
- Ανάπτυξη Recommendation Engine με AI για πρόταση μαθημάτων βάσει προφίλ χρήστη.
- Ενσωμάτωση video streaming με adaptive bitrate για καλύτερη εμπειρία παρακολούθησης.
- Διασύνδεση με Learning Record Stores (LRS) για καταγραφή και ανάλυση μαθησιακής πορείας.
- Πλήρης αυτοματοποίηση DevOps κύκλου με χρήση Kubernetes ή serverless αρχιτεκτονικής.

Οι παραπάνω επεκτάσεις μπορούν να δώσουν σημαντική προστιθέμενη αξία στην πλατφόρμα και να ενισχύσουν την ανταγωνιστικότητά της στην αγορά online εκπαίδευσης.

Αναφορές

[1] UNESCO, "Education in a post-COVID world: nine ideas for public action", 2020. [Ηλεκτρονικό]. Available: https://unesdoc.unesco.org/ark:/48223/pf0000373717.

[2] UDEMY, Udemy Platform, [Ηλεκτρονικό]. Available: https://www.udemy.com/.

[3] COURSERA, Coursera Platform. [Ηλεκτρονικό]. Available: https://www.coursera.org/.

[4] MOODLE, Moodle Project. [Ηλεκτρονικό]. Available: https://moodle.org/.

[5] A. Dennis, B. H. Wixom et al, "Systems Analysis & Design", 2020, 4th Ed., J. Wiley & Sons Inc .

[6] N. Sharma, D. Sawai, 2011. **2011 4th Int.Conf. on Emerging Trends in Engineering & Technology,** 18-20 Nov. 2011, Mauritius, IEEE. [Ηλεκτρονικό]. Available: https://ieeexplore.ieee.org/document/6120543.

[7] E. Ries," The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses", 2011, Penguin Books, London. Available: https://ia800509.us.archive.org/7/items/TheLeanStartupErickRies/The%20Lean%20Startup%20-%20Erick%20Ries.pdf.

[8] H. Portman, "The Minimum Viable Product (MVP) unraveled" *PM World Journal,* Vol. XI, Issue VIII ,2022, . www.pmworldjournal.com

[9] T. DeMarco, "Structured analysis and system specification", 1979, Prentice Hall.

[10] S. Robinson, "Data Flow Diagram (DFD)", *TechTarget*, 2024. [Ηλεκτρονικό]. Available: https://www.techtarget.com/searchdatamanagement/definition/data-flow-diagram-DFD.

[11] L. Poorna, M. Moghul, G. Ananthakrishnan, and H. Arunachalam, “Creating websites using WordPress – A practical approach,” *ResearchGate*, Mar. 2013. [Ηλεκτρονικό]. Available: https://www.researchgate.net/publication/245032512_Creating_websites_using_WordPress_-_A_practical_approach.

[12] *S. Newman, "Building Microservices: Designing Fine-Grained Systems", 2nd Ed. 2021, O'Reilly*

[13] D. Parmenter, *Key Performance Indicators*, Hoboken, NJ: Wiley, 2007.

[14] M. Rozen, “Understanding Page Speed Metrics – Google Lighthouse.” [Ηλεκτρονικό]. Available: https://onlineornot.com/understanding-page-speed-metrics-google-lighthouse

[15] OWASP, “OWASP Top Ten Project.” [Ηλεκτρονικό]. Available: https://owasp.org/www-project-top-ten/

[16] Mattermost, “Cloud Infrastructure Cost KPIs.” [Ηλεκτρονικό]. Available:

https://handbook.mattermost.com/operations/research-and-development/engineering/infrastructure-engineering/cloud-infrastructure-cost-kpis

[17] A. N. Marian Stoica, “Efficiency and Cost-Effectiveness in Agile DevOps with Cloud Computing,” in *Proc. Int. Conf. on Business Excellence*, vol. 18, no. 1, pp. 3453–3456, Jul. 2024.