ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«Σχεδίαση & Ανάπτυξη iOS App για Καλλιτεχνικό Portal»

Του φοιτητή
Αθανάσιου Δ. Χρήστου
Αρ. Μητρώου: 144201

Επιβλέπων
Μιχάλης Σαλαμπάσης
Καθηγητής

ΘΕΣΣΑΛΟΝΙΚΗ ΣΕΠΤΕΜΒΡΙΟΣ 2026

Τίτλος Δ.Ε. Σχεδίαση & Ανάπτυξη IOS App για Καλλιτεχνικό Portal
Κωδικός Δ.Ε. 23149
Ονοματεπώνυμο φοιτητή Αθανάσιος Χρήστου
Ονοματεπώνυμο εισηγητή Μιχάλης Σαλαμπάσης
Ημερομηνία ανάληψης Δ.Ε. 16-03-2023
Ημερομηνία περάτωσης Δ.Ε. 15-07-2026

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 πτυχιακή 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.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Συντομογραφίες

2FA	Two-Factor Authentication (Έλεγχος Ταυτότητας Δύο Παραγόντων)
API	Application Programming Interface
CRUD	Create, Read, Update, Delete
Δ.Ε.	Διπλωματική Εργασία
ΔΙ.ΠΑ.Ε.	Διεθνές Πανεπιστήμιο της Ελλάδος
DTO	Data Transfer Object
EF Core	Entity Framework Core
GDPR	General Data Protection Regulation
HTTP/HTTPS	HyperText Transfer Protocol (Secure)
JSON	JavaScript Object Notation
JWT	JSON Web Token
ORM	Object-Relational Mapper
OTP	One-Time Password
REST	REpresentational State Transfer
SDK	Software Development Kit
SUS	System Usability Scale
UI	User Interface (Διεπαφή Χρήστη)
UX	User Experience (Εμπειρία Χρήστη)

Πρόλογος

Η ανάπτυξη της παρούσας εφαρμογής, η οποία αφορά την ανάκτηση και παρουσίαση δεδομένων θεατρικών παραστάσεων, αποτελεί εξέλιξη μιας προγενέστερης εργασίας, ενσωματώνοντας βελτιώσεις τόσο σε επίπεδο λειτουργικότητας όσο και στη συνολική εμπειρία του χρήστη. Το έργο αυτό βασίστηκε σε μια συνεργατική προσέγγιση, όπου κάθε μέλος της ομάδας ανέλαβε την υλοποίηση διακριτών τμημάτων του συστήματος, συμβάλλοντας στη δημιουργία μιας ολοκληρωμένης και αποδοτικής λύσης.

Η σύνθεση των επιμέρους εργασιών αποτέλεσε μια πολύτιμη εμπειρία εκμάθησης, εστιάζοντας όχι μόνο στην ανάπτυξη λογισμικού μεγάλης κλίμακας αλλά και στη βελτιστοποίηση της διαδραστικότητας και της προσωποποιημένης αλληλεπίδρασης με τον χρήστη. Πέρα από την απλή ανάκτηση και παρουσίαση δεδομένων, η εφαρμογή προσφέρει δυνατότητες όπως η διαχείριση προφίλ, η αγορά μονάδων, η ανάρτηση περιεχομένου και άλλες λειτουργίες που ενισχύουν τη συνολική εμπειρία του χρήστη.

Για την υλοποίηση της εφαρμογής, κρίθηκε αναγκαία η εμβάθυνση σε σύγχρονες τεχνολογίες και εργαλεία ανάπτυξης λογισμικού. Ιδιαίτερη σημασία δόθηκε στη χρήση ενός ευέλικτου και αποδοτικού framework, το οποίο επιτρέπει τη δημιουργία σύγχρονων διεπαφών χρήστη και τη διατήρηση ενός ομοιογενούς περιβάλλοντος αλληλεπίδρασης. Η επιλογή αυτή, σε συνδυασμό με τη συνεχή αναβάθμιση της αρχικής εργασίας, προσφέρει νέες επαγγελματικές προοπτικές, καθώς διευρύνει τις γνώσεις και τις δεξιότητες που απαιτούνται για την ανάπτυξη σύγχρονων και πολυλειτουργικών εφαρμογών.

Περίληψη

Αντικείμενο της παρούσας πτυχιακής εργασίας είναι η αναβάθμιση και βελτίωση μιας ήδη υπάρχουσας εφαρμογής, η οποία αφορά την ανάκτηση και παρουσίαση δεδομένων θεατρικών παραστάσεων. Στόχος της αναβάθμισης είναι η προσαρμογή της εφαρμογής στα νέα δεδομένα και στις σύγχρονες απαιτήσεις των χρηστών, βελτιώνοντας τόσο τη λειτουργικότητα όσο και τη συνολική εμπειρία χρήστη.

Για την ανάπτυξη της εφαρμογής χρησιμοποιήθηκε το **Flutter framework** και η γλώσσα προγραμματισμού **Dart**, ώστε να διασφαλιστεί η δημιουργία μιας σύγχρονης, αποδοτικής και καλαίσθητης διεπαφής. Το Flutter επιλέχθηκε λόγω της ευελιξίας του, της δυνατότητας ανάπτυξης **cross-platform εφαρμογών**, καθώς και της εύκολης προσαρμογής του στις απαιτήσεις της εργασίας.

Η ανάπτυξη της εφαρμογής βασίστηκε τόσο σε προκαθορισμένες προδιαγραφές που δόθηκαν, όσο και σε επιπλέον λειτουργικότητες που κρίθηκαν απαραίτητες για τη βελτίωση της συνολικής εμπειρίας χρήστη.

Επιπλέον, ιδιαίτερη έμφαση δόθηκε στο **optimization** της εφαρμογής, διασφαλίζοντας υψηλή απόδοση, γρήγορους χρόνους απόκρισης και βέλτιστη διαχείριση δεδομένων. Έγινε βελτιστοποίηση του κώδικα, περιορισμός της κατανάλωσης πόρων και εφαρμογή τεχνικών caching και state management, ώστε η εφαρμογή να είναι πιο αποδοτική και να προσφέρει μια ομαλή και απρόσκοπτη εμπειρία χρήστη.

Μέσα από αυτή τη διαδικασία, η εργασία δεν περιορίστηκε μόνο στην τεχνική υλοποίηση, αλλά επεκτάθηκε στη **βελτιστοποίηση της διαδραστικότητας**, στη βελτίωση της οπτικής παρουσίασης και στην ενσωμάτωση σύγχρονων τεχνολογικών προτύπων, καθιστώντας την εφαρμογή πιο αποδοτική και φιλική προς τον χρήστη.

Abstract

The objective of this thesis is the upgrade and enhancement of an existing application that focuses on retrieving and presenting data related to theatrical performances. The purpose of this upgrade is to adapt the application to new data requirements and modern user demands, improving both its functionality and overall user experience.

For the development of the application, the **Flutter framework** and the **Dart programming language** were utilized to ensure the creation of a modern, efficient, and visually appealing user interface. Flutter was chosen due to its flexibility, its **cross-platform development capabilities**, and its adaptability to the specific requirements of this project.

The development process was based on predefined specifications as well as additional functionalities that were deemed necessary to enhance the overall user experience.
Furthermore, special emphasis was placed on the **optimization** of the application, ensuring **high performance, fast response times, and efficient data management**. Code optimization was carried out, along with resource consumption reduction and the implementation of **caching and state management techniques**, making the application more efficient and capable of delivering a smooth and seamless user experience.

Through this process, the project extended beyond mere technical implementation, focusing on **interaction optimization**, improving visual presentation, and integrating modern technological standards, ultimately making the application more efficient and user-friendly.

Ευχαριστίες

Θα ήθελα, αρχικά, να εκφράσω τη βαθιά μου ευγνωμοσύνη στους γονείς μου και στα αδέρφια μου, οι οποίοι στάθηκαν δίπλα μου σε όλη τη διάρκεια των σπουδών μου. Με υπομονή, στήριξη και αγάπη, μου έδωσαν όλα τα απαραίτητα εφόδια για να ακολουθήσω αυτό το ταξίδι και να φτάσω μέχρι την ολοκλήρωσή του.

Ιδιαίτερες ευχαριστίες οφείλω στους καθηγητές μου, οι οποίοι μέσα από τη γνώση, την καθοδήγηση και την εμπειρία τους συνέβαλαν ουσιαστικά στη διαμόρφωση της ακαδημαϊκής και επαγγελματικής μου πορείας.

Τέλος, θα ήθελα να αφιερώσω αυτή την εργασία στη μνήμη του παππού μου, ενός ανθρώπου στον οποίο είχα ιδιαίτερη αδυναμία και που κατείχε ξεχωριστή θέση στη ζωή μου. Ίσως τελικά η ολοκλήρωση αυτής της εργασίας να αποτελεί και έναν τελευταίο διάλογο μαζί του.

Αυτή η στιγμή είναι και για εκείνον. Μια υπόσχεση που έγινε πράξη…

1. Εισαγωγή

Στη σύγχρονη ψηφιακή εποχή, ο τρόπος με τον οποίο το κοινό ανακαλύπτει, προσεγγίζει και αλληλεπιδρά με τον πολιτισμό υφίσταται μια διαρκή μεταμόρφωση. Το θέατρο, ως μια από τις αρχαιότερες και πιο ζωντανές μορφές τέχνης, καλείται να αγκαλιάσει τις νέες τεχνολογίες όχι απλώς ως μέσο προβολής, αλλά ως γέφυρα για τη δημιουργία μιας πιο άμεσης και ουσιαστικής σχέσης με το θεατρόφιλο κοινό. Η παρούσα διπλωματική εργασία τοποθετείται στο επίκεντρο αυτής της σύγκλισης, διερευνώντας τη διαδικασία σχεδίασης, ανάπτυξης και υλοποίησης μιας ολοκληρωμένης ψηφιακής πλατφόρμας, αφιερωμένης στον χώρο του ελληνικού θεάτρου [1].

1.1 Σκοπός και Στόχοι της Εργασίας

Η ανάπτυξη πληροφοριακών συστημάτων απαιτεί τόσο τεχνική συνέπεια όσο και ομαδική συνεργασία, ειδικά όταν πρόκειται για πολυδιάστατα έργα που περιλαμβάνουν διαφορετικές τεχνολογικές πλατφόρμες. Η συγκεκριμένη πτυχιακή εργασία αποτελεί αναπόσπαστο μέρος μιας συλλογικής προσπάθειας που πραγματοποιήθηκε από μια ομάδα φοιτητών, με στόχο τη δημιουργία ενός ολοκληρωμένου συστήματος διαχείρισης και παρουσίασης θεατρικών παραστάσεων.

Αρχικά, η διαδικασία ξεκίνησε με τη συλλογή δεδομένων μέσω web scraping, όπου αντλήθηκαν πληροφορίες σχετικά με θεατρικές παραστάσεις. Τα δεδομένα αυτά αποθηκεύτηκαν σε μια βάση δεδομένων, δημιουργώντας έτσι την ανάγκη για ένα εργαλείο που θα επιτρέπει την παρουσίασή τους με κατανοητό και ελκυστικό τρόπο. Για την υλοποίηση αυτού του στόχου, αναπτύχθηκαν τρεις ξεχωριστές εφαρμογές: μία για web, μία για Android και μία για iOS, οι οποίες αξιοποιούν τα ίδια δεδομένα αλλά με διαφορετικές προσεγγίσεις στη σχεδίαση και τη διαχείριση περιεχομένου.

Για να εξασφαλιστεί η ομοιομορφία και η συνεκτικότητα ανάμεσα στις τρεις αυτές εφαρμογές, κρίθηκε απαραίτητη η δημιουργία ενός API, το οποίο θα διαχειρίζεται την επικοινωνία μεταξύ της βάσης δεδομένων και των εφαρμογών. Χρησιμοποιώντας JSON, τα δεδομένα διατίθενται σε όλες τις πλατφόρμες, επιτρέποντας έτσι τη δυναμική και εύκολη ενημέρωσή τους. Η ομάδα που ασχολήθηκε με το API είχε συνεχή επικοινωνία με όσους ανέπτυσσαν τις εφαρμογές, ώστε τα endpoints να ανταποκρίνονται στις ανάγκες παρουσίασης των δεδομένων και να παρέχουν τη βέλτιστη εμπειρία χρήστη.
Ένας βασικός στόχος της παρούσας εργασίας ήταν η δημιουργία ενός συστήματος απλής και άμεσης πρόσβασης στην πληροφορία, ώστε οι χρήστες να μπορούν εύκολα να αναζητούν και να παρακολουθούν θεατρικές παραστάσεις. Για να επιτευχθεί αυτό, αρχικά σχεδιάστηκαν οθόνες και λειτουργίες που θα χρησιμοποιούνταν στις εφαρμογές, ώστε να διασφαλιστεί η συνέπεια στη δομή και η βελτιστοποίηση της εμπειρίας χρήστη. Αυτό το βήμα συνέβαλε όχι μόνο στη σωστή οργάνωση της πληροφορίας αλλά και στη διευκόλυνση της συνεργασίας μεταξύ των ομάδων ανάπτυξης.

Όσον αφορά την παρούσα πτυχιακή εργασία, η ανάπτυξη της εφαρμογής iOS αποτέλεσε το κύριο αντικείμενο μελέτης. Στην πραγματικότητα, η συγκεκριμένη εφαρμογή αποτελεί

αναβαθμισμένη έκδοση μιας προηγούμενης εφαρμογής που είχε αναπτυχθεί στο πλαίσιο παλαιότερης πτυχιακής εργασίας. Η προηγούμενη έκδοση παρείχε τις βασικές λειτουργίες αλλά παρουσίαζε περιορισμούς τόσο στη σχεδίαση όσο και στην απόδοσή της. Έτσι, η νέα αυτή εκδοχή δεν αποτελεί απλώς βελτίωση της υπάρχουσας εφαρμογής, αλλά μια πλήρη ανασχεδίαση και αναβάθμιση, με στόχο να προσφέρει σύγχρονη εμπειρία χρήστη, καλύτερη απόδοση και διευρυμένες δυνατότητες.

Μετά από έρευνα και αξιολόγηση διαφόρων εργαλείων, επιλέχθηκε το Flutter framework ως η βέλτιστη λύση για την ανάπτυξη της εφαρμογής. Η επιλογή του Flutter βασίστηκε στο γεγονός ότι προσέφερε σημαντικά πλεονεκτήματα, όπως [2], [3]:

- Σύγχρονο και ελκυστικό UI design, που βελτιώνει την εμπειρία χρήστη.
- Υποστήριξη cross-platform ανάπτυξης, επιτρέποντας την εύκολη προσαρμογή του κώδικα και σε άλλες πλατφόρμες.
- Γρήγορη ανάπτυξη και υψηλές επιδόσεις, που διευκόλυναν την υλοποίηση των απαιτούμενων λειτουργιών.
- Βελτιστοποιημένη διαχείριση δεδομένων και καλύτερη απόκριση σε σχέση με την προηγούμενη εφαρμογή.

Η εφαρμογή iOS σχεδιάστηκε με γνώμονα τη λειτουργικότητα και την ευχρηστία, παρέχοντας έναν εύκολο τρόπο στους χρήστες να πλοηγηθούν στα δεδομένα των θεατρικών παραστάσεων [4].

Συμπερασματικά, η παρούσα εργασία δεν εστιάζει απλώς στην ανάπτυξη μιας εφαρμογής αλλά αποτελεί μέρος μιας ευρύτερης προσπάθειας για τη δημιουργία ενός πληροφοριακού συστήματος που εξυπηρετεί πολλαπλές πλατφόρμες. Παράλληλα, με την αναβάθμιση της υπάρχουσας εφαρμογής, επιτεύχθηκε η βελτίωση της απόδοσης και της αισθητικής, καθιστώντας την εφαρμογή πιο μοντέρνα, λειτουργική και εύχρηστη. Μέσα από τη συνεργασία, τη σωστή αξιοποίηση εργαλείων και την έρευνα, επιτεύχθηκε η ανάπτυξη μιας εφαρμογής που ανταποκρίνεται στις απαιτήσεις των χρηστών, διατηρώντας παράλληλα τεχνική συνέπεια, υψηλή αισθητική και λειτουργικότητα.

1.2 Σημασία της Εργασίας

Η σημασία της παρούσας διπλωματικής εργασίας είναι πολυεπίπεδη, καθώς το παραδοτέο της έργο φιλοδοξεί να επιφέρει θετικό αντίκτυπο τόσο στον πολιτιστικό κλάδο, όσο και στο πεδίο της εφαρμοσμένης πληροφορικής.
Πρωτίστως, η εργασία απαντά σε μια υπαρκτή ανάγκη του ελληνικού θεατρικού οικοσυστήματος για ψηφιακό μετασχηματισμό και κεντρικοποίηση της πληροφορίας. Στο σημερινό τοπίο, τα δεδομένα που αφορούν θεατρικές παραστάσεις, συντελεστές και καλλιτεχνικούς χώρους είναι κατακερματισμένα σε πλήθος ετερογενών πηγών, όπως ιστοσελίδες θεάτρων, πλατφόρμες έκδοσης εισιτηρίων και μέσα κοινωνικής δικτύωσης. Αυτός ο κατακερματισμός καθιστά την αναζήτηση και τη συνολική επισκόπηση μια χρονοβόρα και συχνά αναποτελεσματική διαδικασία για το κοινό. Η πλατφόρμα που αναπτύχθηκε έρχεται να καλύψει αυτό το κενό, προσφέροντας ένα ενιαίο, αξιόπιστο και φιλικό προς τον χρήστη ψηφιακό αποθετήριο, το οποίο συγκεντρώνει και διασυνδέει τις σχετικές πληροφορίες [5].

Πέρα από την αξία της για το θεατρόφιλο κοινό, η πλατφόρμα ενισχύει την ψηφιακή παρουσία των ίδιων των καλλιτεχνών και των επαγγελματιών του χώρου. Μέσω καινοτόμων λειτουργιών, όπως το σύστημα διεκδίκησης προφίλ (claiming system), οι δημιουργοί μετατρέπονται από παθητικοί αποδέκτες σε ενεργούς διαμορφωτές της ψηφιακής τους ταυτότητας, αποκτώντας τον έλεγχο των πληροφοριών που τους αφορούν. Με αυτόν τον τρόπο, η εργασία συμβάλλει στην προώθηση του πολιτιστικού προϊόντος, αξιοποιώντας την τεχνολογία ως γέφυρα επικοινωνίας μεταξύ της τέχνης του θεάτρου και του σύγχρονου, ψηφιακά εγγράμματου κοινού. Ιδιαίτερη έμφαση δίνεται στην προσέγγιση των νεότερων γενεών, οι οποίες χρησιμοποιούν κατά κύριο λόγο τις κινητές συσκευές για την ενημέρωση και την ψυχαγωγία τους, καθιστώντας την εφαρμογή ένα οικείο και προσιτό μέσο ανακάλυψης της θεατρικής τέχνης.
Από τεχνολογικής σκοπιάς, η εργασία αποτελεί μια ολοκληρωμένη μελέτη περίπτωσης (case study) για την ανάπτυξη σύγχρονων, cross-platform εφαρμογών. Η στρατηγική επιλογή του framework Flutter και της γλώσσας προγραμματισμού Dart προσδίδει σημαντική αξία στο εγχείρημα, καθώς επιτρέπει την ανάπτυξη μιας ενιαίας βάσης κώδικα (single codebase) που λειτουργεί απρόσκοπτα τόσο σε περιβάλλον iOS όσο και Android. Αυτή η προσέγγιση όχι μόνο μειώνει δραστικά τον χρόνο και το κόστος ανάπτυξης και συντήρησης, αλλά διασφαλίζει και μια συνεπή εμπειρία χρήστη σε όλες τις πλατφόρμες. Η αρχιτεκτονική του συστήματος, που συνδυάζει την ευελιξία του Flutter στο frontend με τη στιβαρότητα του ASP.NET Core στο backend, καταδεικνύει την πρακτική εφαρμογή σύγχρονων αρχών λογισμικού για τη δημιουργία επεκτάσιμων και ασφαλών ψηφιακών λύσεων στον πολιτιστικό τομέα [6].

1.3 Μεθοδολογία

Η προσέγγιση που ακολουθήθηκε για την υλοποίηση του έργου βασίστηκε σε μια ευέλικτη, επαναληπτική μεθοδολογία ανάπτυξης, η οποία μπορεί να συνοψιστεί στα παρακάτω στάδια:

1. **Ανάλυση Απαιτήσεων και Σχεδιασμός:** Το αρχικό στάδιο περιελάμβανε τον καθορισμό των λειτουργικών και μη-λειτουργικών απαιτήσεων του συστήματος. Πραγματοποιήθηκε ο σχεδιασμός της αρχιτεκτονικής client-server, η μοντελοποίηση του σχήματος της βάσης δεδομένων (Entity-Relationship Diagram) και ο καθορισμός των βασικών endpoints του API.
2. **Ανάπτυξη Backend:** Το επόμενο βήμα ήταν η υλοποίηση της λογικής του εξυπηρετητή. Αναπτύχθηκαν τα controllers, οι υπηρεσίες (services) και τα αποθετήρια (repositories) που απαρτίζουν το ASP.NET Core API. Ιδιαίτερη έμφαση δόθηκε στη δημιουργία μηχανισμών ασφαλείας (authentication/authorization) και στην επικοινωνία με τη βάση δεδομένων μέσω του Entity Framework Core.
3. **Ανάπτυξη Frontend:** Παράλληλα με το backend, ξεκίνησε η υλοποίηση της εφαρμογής για κινητά σε Flutter. Δημιουργήθηκαν οι οθόνες (pages), τα widgets και οι υπηρεσίες (services) που διαχειρίζονται την επικοινωνία με το API, την κατάσταση της εφαρμογής (state management) και την παρουσίαση των δεδομένων στον χρήστη.
4. **Ολοκλήρωση και Δοκιμές (Integration & Testing):** Σε αυτό το στάδιο, τα δύο μέρη του συστήματος (frontend και backend) συνδέθηκαν και δοκιμάστηκε η ορθή συνεργασία τους. Πραγματοποιήθηκε η ενσωμάτωση των εξωτερικών υπηρεσιών

και διεξήχθησαν εκτενείς έλεγχοι (manual & integration testing) για την εξακρίβωση της ορθής λειτουργίας των σεναρίων χρήσης, από την εγγραφή ενός χρήστη μέχρι την υποβολή ενός αιτήματος "claim".

5. **Βελτιστοποίηση και Ανάδραση:** Η ανάπτυξη ακολούθησε έναν κυκλικό χαρακτήρα, όπου η ανάδραση από τις δοκιμές οδηγούσε σε βελτιώσεις και προσθήκες νέων λειτουργιών, διασφαλίζοντας τη συνεχή εξέλιξη του συστήματος.

1.4 Δομή της Εργασίας

Η παρούσα διπλωματική εργασία είναι οργανωμένη στα ακόλουθα κεφάλαια, προκειμένου να παρουσιάσει με σαφήνεια και πληρότητα όλα τα στάδια του έργου:

- **Κεφάλαιο 2 - Θεωρητικό Υπόβαθρο:** Αναλύονται οι τεχνολογίες και οι έννοιες που αποτελούν τη βάση της υλοποίησης. Εξετάζεται ο ρόλος των mobile εφαρμογών στον πολιτιστικό τομέα, συγκρίνονται οι τεχνολογίες ανάπτυξης εφαρμογών και διερευνώνται οι αρχές σχεδίασης UI/UX σε πολιτιστικές εφαρμογές.
- **Κεφάλαιο 3 - Ανάλυση Απαιτήσεων:** Παρουσιάζονται αναλυτικά οι λειτουργικές και μη-λειτουργικές απαιτήσεις του συστήματος, τα προφίλ των χρηστών και τα σενάρια χρήσης που καθοδήγησαν τον σχεδιασμό.
- **Κεφάλαιο 4 - Σχεδίαση Συστήματος:** Περιγράφεται σε βάθος η αρχιτεκτονική του συστήματος, η δομή του API, το σχήμα της βάσης δεδομένων, οι μηχανισμοί ασφαλείας και η σχεδίαση του UI/UX, συνοδευόμενα από τα αντίστοιχα διαγράμματα.
- **Κεφάλαιο 5 - Υλοποίηση:** Αναλύεται η τεχνική υλοποίηση του έργου, παρουσιάζοντας τη δομή του κώδικα, παραδείγματα από την επικοινωνία με το API, τις δυσκολίες που προέκυψαν και τις λύσεις που δόθηκαν.
- **Κεφάλαιο 6 - Αξιολόγηση & Testing:** Περιγράφεται η μεθοδολογία ελέγχου που ακολουθήθηκε, παρατίθενται τα σενάρια δοκιμών και παρουσιάζονται τα αποτελέσματα της αξιολόγησης του συστήματος.
- **Κεφάλαιο 7 - Συμπεράσματα & Μελλοντική Εργασία:** Συνοψίζονται τα βασικά επιτεύγματα της εργασίας, αναδεικνύεται η συμβολή της, καταγράφονται οι περιορισμοί και προτείνονται ιδέες για μελλοντικές επεκτάσεις και βελτιώσεις.

2. Θεωρητικό Υπόβαθρο

Η επιτυχής υλοποίηση ενός σύγχρονου πληροφοριακού συστήματος προϋποθέτει τη βαθιά κατανόηση του θεωρητικού πλαισίου στο οποίο εντάσσεται. Το παρόν κεφάλαιο αποσκοπεί στην παρουσίαση και ανάλυση των θεμελιωδών εννοιών και τεχνολογιών που αποτέλεσαν τη βάση για τον σχεδιασμό και την ανάπτυξη της εφαρμογής «Θεατρικό». Μέσα από την επισκόπηση της βιβλιογραφίας και την ανάλυση των τρεχουσών τάσεων, το κεφάλαιο αυτό χτίζει τη γνωσιακή υποδομή που δικαιολογεί τις αρχιτεκτονικές και τεχνολογικές επιλογές της παρούσας διπλωματικής εργασίας. Η ανάλυση θα κινηθεί σε τρεις βασικούς άξονες: τον ρόλο των mobile εφαρμογών στον πολιτιστικό τομέα, τις διαθέσιμες τεχνολογίες ανάπτυξης και τις κρίσιμες αρχές σχεδιασμού της εμπειρίας χρήστη (UI/UX) για πολιτιστικό περιεχόμενο.

2.1 Εισαγωγή

Ο τομέας της ανάπτυξης εφαρμογών για κινητές συσκευές (mobile application development) αποτελεί έναν από τους πιο δυναμικούς και ταχέως εξελισσόμενους κλάδους της σύγχρονης πληροφορικής. Στη σημερινή ψηφιακή πραγματικότητα, οι απαιτήσεις των χρηστών έχουν ωριμάσει, διαμορφώνοντας ένα περιβάλλον όπου η επιτυχία μιας εφαρμογής δεν κρίνεται μόνο από τη λειτουργικότητά της, αλλά και από ένα σύνολο ποιοτικών χαρακτηριστικών. Οι χρήστες αναμένουν εφαρμογές που είναι ταυτόχρονα γρήγορες σε απόκριση, αισθητικά άρτιες, διαισθητικές στη χρήση και απόλυτα σταθερές.

Η παρούσα διπλωματική εργασία εστιάζει στην ανάπτυξη μιας ολοκληρωμένης ψηφιακής πλατφόρμας που λειτουργεί ως πύλη στον κόσμο των ελληνικών θεατρικών παραστάσεων. Η εφαρμογή για κινητές συσκευές, αν και αρχικά σχεδιάστηκε με γνώμονα το λειτουργικό σύστημα iOS, υλοποιήθηκε με τεχνολογίες πολλαπλών πλατφορμών (cross-platform) ώστε να διασφαλιστεί η ευρεία προσβασιμότητά της.

Το παρόν κεφάλαιο αποσκοπεί στη θεμελίωση του θεωρητικού υποβάθρου πάνω στο οποίο στηρίχθηκαν οι σχεδιαστικές και τεχνολογικές αποφάσεις του έργου. Η ανάλυση που ακολουθεί καλύπτει τους κεντρικούς άξονες που διέπουν τη δημιουργία σύγχρονων πολιτιστικών εφαρμογών, εξετάζοντας:

- Την ιστορική εξέλιξη και τον αυξανόμενο ρόλο των mobile εφαρμογών στον πολιτιστικό τομέα.
- Τις κυρίαρχες τεχνολογίες και μεθοδολογίες ανάπτυξης εφαρμογών, με έμφαση στη σύγκριση μεταξύ Native και Cross-Platform προσεγγίσεων και την αιτιολόγηση της επιλογής του Flutter.
- Τις θεμελιώδεις αρχές σχεδιασμού της διεπαφής και της εμπειρίας χρήστη (UI/UX) που είναι κρίσιμες για εφαρμογές πολιτιστικού περιεχομένου.
- Την ανάλυση επιλεγμένων μελετών περίπτωσης (case studies) από τον διεθνή και εγχώριο χώρο, με σκοπό την άντληση συμπερασμάτων και καλών πρακτικών.

2.2 Mobile Εφαρμογές στον Πολιτισμό

Η τελευταία δεκαετία υπήρξε καθοριστική για τον ψηφιακό μετασχηματισμό των πολιτιστικών οργανισμών. Η εκθετική διείσδυση των έξυπνων κινητών τηλεφώνων (smartphones) στην καθημερινότητα ώθησε μουσεία, θέατρα, φεστιβάλ και άλλους πολιτιστικούς φορείς να αναζητήσουν νέους, πιο άμεσους διαύλους επικοινωνίας με το κοινό τους. Σε αυτό το πλαίσιο, οι εφαρμογές για κινητά εξελίχθηκαν από ένα απλό συμπληρωματικό εργαλείο σε έναν στρατηγικό πυλώνα για την προσέγγιση, την ενημέρωση και την ενίσχυση της πολιτιστικής εμπειρίας [1].

Ιστορική Αναδρομή και Εξέλιξη

Στις αρχές της δεκαετίας του 2010, η ψηφιακή παρουσία των περισσότερων πολιτιστικών οργανισμών περιοριζόταν σε στατικές ιστοσελίδες, οι οποίες συχνά δεν ήταν βελτιστοποιημένες για προβολή σε κινητές συσκευές. Οι πρώτες εφαρμογές που εμφανίστηκαν λειτουργούσαν κυρίως ως ψηφιακά αντίγραφα των ιστοσελίδων, προσφέροντας βασικές πληροφορίες όπως ωράρια λειτουργίας, τοποθεσίες και προγράμματα εκδηλώσεων.
Σήμερα, η εικόνα έχει αλλάξει ριζικά. Οι σύγχρονες πολιτιστικές εφαρμογές έχουν μετατραπεί σε ολοκληρωμένες πύλες περιεχομένου και αλληλεπίδρασης. Προσφέρουν δυναμικές λειτουργίες όπως εξατομικευμένες προτάσεις, δυνατότητες κοινωνικής δικτύωσης (π.χ. κοινοποίηση εκδηλώσεων), αγορά ηλεκτρονικών εισιτηρίων και ασφαλείς πληρωμές. Η τάση αυτή καταδεικνύει μια σαφή μετατόπιση: οι εφαρμογές δεν αποτελούν πλέον απλά εργαλεία πληροφόρησης, αλλά αναπόσπαστα μέσα για την αναβάθμιση και τον εμπλουτισμό της συνολικής πολιτιστικής εμπειρίας του χρήστη, πριν, κατά τη διάρκεια και μετά την επίσκεψή του [5].

Εφαρμογές ανά Πολιτιστικό Τομέα

- **Μουσεία και Εκθεσιακοί Χώροι:** Οι εφαρμογές προσφέρουν εικονικές περιηγήσεις, διαδραστικούς χάρτες και ηχητικές ξεναγήσεις. Αξιοποιώντας τεχνολογίες όπως η Επαυξημένη Πραγματικότητα (Augmented Reality), μπορούν να προβάλλουν επιπρόσθετες πληροφορίες πάνω στα εκθέματα, ζωντανεύοντας την ιστορία τους [7].
- **Θέατρα και Χώροι Παραστατικών Τεχνών:** Οι θεατρικές εφαρμογές διευκολύνουν την αγορά και διαχείριση εισιτηρίων, παρέχουν πληροφορίες για τις παραστάσεις (συντελεστές, περίληψη, κριτικές) και προσφέρουν πρόσβαση σε ψηφιακό αρχειακό υλικό, όπως προγράμματα παλαιότερων παραστάσεων και φωτογραφίες.
- **Φεστιβάλ:** Για διοργανώσεις μεγάλης κλίμακας, οι εφαρμογές είναι απαραίτητες για τη διαχείριση του πολύπλοκου προγράμματος. Επιτρέπουν στους χρήστες να δημιουργούν το προσωπικό τους πρόγραμμα, να λαμβάνουν ειδοποιήσεις σε πραγματικό χρόνο για αλλαγές και να πλοηγούνται στους χώρους του φεστιβάλ μέσω ενσωματωμένων χαρτών.

2.3 Τεχνολογίες Ανάπτυξης Εφαρμογών

Η επιλογή της κατάλληλης τεχνολογικής πλατφόρμας αποτελεί μια από τις πιο κρίσιμες αποφάσεις στη διαδικασία δημιουργίας μιας mobile εφαρμογής, καθώς επηρεάζει άμεσα την απόδοση, το κόστος ανάπτυξης, τον χρόνο παράδοσης και τη μελλοντική συντηρησιμότητα του έργου. Οι κυρίαρχες προσεγγίσεις κατηγοριοποιούνται σε Native και Cross-Platform Development [2].

- **Native Development:** Η προσέγγιση αυτή περιλαμβάνει τη συγγραφή κώδικα ξεχωριστά για κάθε λειτουργικό σύστημα, χρησιμοποιώντας τις επίσημες γλώσσες και εργαλεία που παρέχει ο κάθε κατασκευαστής.
    - **iOS (Swift, Objective-C):** Η ανάπτυξη για το οικοσύστημα της Apple προσφέρει τη μέγιστη δυνατή απόδοση, άμεση πρόσβαση στις τελευταίες λειτουργίες του λειτουργικού συστήματος και μια εμπειρία χρήστη που εναρμονίζεται πλήρως με τις σχεδιαστικές συμβάσεις της πλατφόρμας [8].
    - **Android (Kotlin, Java):** Αντίστοιχα, η ανάπτυξη για Android παρέχει βέλτιστη απόδοση και πλήρη αξιοποίηση των δυνατοτήτων του συστήματος. Το κύριο μειονέκτημα της native προσέγγισης είναι η ανάγκη για διατήρηση δύο διαφορετικών βάσεων κώδικα, γεγονός που αυξάνει το κόστος και την πολυπλοκότητα.
- **Cross-Platform Development:** Η μεθοδολογία αυτή επιτρέπει στους προγραμματιστές να γράφουν τον κώδικα μία φορά και να τον μεταγλωττίζουν (compile) για να λειτουργεί τόσο σε iOS όσο και σε Android [6].
    - **Flutter (Dart):** Το Flutter, το οποίο αναπτύσσεται από την Google, έχει αναδειχθεί σε μία από τις κορυφαίες λύσεις cross-platform. Επιλέχθηκε για την παρούσα εργασία λόγω των σημαντικών πλεονεκτημάτων που προσφέρει [3]:
        - **Ενιαία Βάση Κώδικα (Single Codebase):** Μειώνει δραστικά τον χρόνο και την προσπάθεια που απαιτείται για την ανάπτυξη, τον έλεγχο και τη συντήρηση της εφαρμογής.
        - **Υψηλή Απόδοση:** Το Flutter μεταγλωττίζεται απευθείας σε native κώδικα ARM, επιτυγχάνοντας επιδόσεις που προσεγγίζουν αυτές των native εφαρμογών [9].
        - **Εκφραστικό και Ευέλικτο UI:** Η αρχιτεκτονική του, βασισμένη σε widgets, επιτρέπει τη δημιουργία σύνθετων και αισθητικά άρτιων γραφικών περιβαλλόντων που παραμένουν συνεπή σε όλες τις πλατφόρμες.
        - **Ταχύτατος Κύκλος Ανάπτυξης:** Η λειτουργία "Hot Reload" επιτρέπει στους προγραμματιστές να βλέπουν τις αλλαγές στον κώδικα να εφαρμόζονται στην εφαρμογή σχεδόν ακαριαία, επιταχύνοντας δραματικά τη διαδικασία σχεδιασμού και αποσφαλμάτωσης.
        - **Ισχυρή Κοινότητα και Υποστήριξη:** Ως προϊόν της Google, το Flutter απολαμβάνει συνεχείς ενημερώσεις και υποστηρίζεται από μια μεγάλη και ενεργή κοινότητα προγραμματιστών.
    - Άλλες δημοφιλείς τεχνολογίες στην ίδια κατηγορία περιλαμβάνουν το **React Native** (βασισμένο σε JavaScript) και το **Xamarin** (βασισμένο σε C#).

2.4 Σχεδίαση UI/UX σε Πολιτιστικές Εφαρμογές

Στο πλαίσιο μιας πολιτιστικής εφαρμογής, η σχεδίαση της Διεπαφής Χρήστη (User Interface - UI) και της Εμπειρίας Χρήστη (User Experience - UX) αποκτά ακόμα μεγαλύτερη σημασία. Το UI αφορά την οπτική εμφάνιση και τη διάταξη των στοιχείων με τα οποία αλληλεπιδρά ο χρήστης, ενώ το UX περιγράφει τη συνολική αίσθηση και τον βαθμό ικανοποίησης του χρήστη από αυτή την αλληλεπίδραση. Μια επιτυχημένη σχεδίαση πρέπει να εναρμονίζει την αισθητική αρτιότητα με την αβίαστη λειτουργικότητα, καθιστώντας την πλούσια πολιτιστική πληροφορία προσιτή και ελκυστική.

Βασικές Αρχές Σχεδιασμού

- **Απλότητα και Σαφήνεια:** Η πλοήγηση πρέπει να είναι διαισθητική. Ο χρήστης οφείλει να μπορεί να εντοπίσει την πληροφορία που αναζητά (π.χ. μια παράσταση, έναν συντελεστή) με τις ελάχιστες δυνατές ενέργειες, χωρίς να έρχεται αντιμέτωπος με υπερφορτωμένες οθόνες [10].
- **Οπτική Συνέπεια:** Η χρήση μιας συνεκτικής οπτικής γλώσσας (χρώματα, τυπογραφία, εικονίδια) σε όλες τις οθόνες της εφαρμογής ενισχύει την αναγνωρισιμότητα και δημιουργεί μια αίσθηση οικειότητας και επαγγελματισμού [11].
- **Προσβασιμότητα (Accessibility):** Ο σχεδιασμός πρέπει να λαμβάνει υπόψη τους χρήστες με ειδικές ανάγκες. Αυτό περιλαμβάνει την υποστήριξη για προγράμματα ανάγνωσης οθόνης (screen readers), τη δυνατότητα προσαρμογής του μεγέθους της γραμματοσειράς και τη χρήση χρωμάτων με επαρκή αντίθεση (contrast).
- **Εξατομίκευση (Personalization):** Η δυνατότητα της εφαρμογής να προσαρμόζει το περιεχόμενο στις προτιμήσεις του χρήστη (π.χ. προτείνοντας παραστάσεις με βάση το ιστορικό του) μπορεί να αυξήσει σημαντικά την αφοσίωσή του.

Προκλήσεις στον Σχεδιασμό Πολιτιστικών Εφαρμογών

- **Διαχείριση Πλούσιου Περιεχομένου:** Η κύρια πρόκληση είναι η οργάνωση και παρουσίαση μεγάλου όγκου δεδομένων (π.χ. μακροσκελείς λίστες συντελεστών, αναλυτικές περιγραφές έργων) με τρόπο που να μην κουράζει ή αποπροσανατολίζει τον χρήστη.
- **Ενσωμάτωση Πολυμέσων:** Η απρόσκοπτη ενσωμάτωση εικόνων, βίντεο και ήχου υψηλής ποιότητας, χωρίς να υποβαθμίζεται η απόδοση της εφαρμογής, είναι κρίσιμης σημασίας.
- **Αρχιτεκτονική της Πληροφορίας:** Ο σχεδιασμός μιας λογικής και κατανοητής ιεραρχίας για τη σύνδεση ετερογενών οντοτήτων (παραστάσεις, πρόσωπα, χώροι, ημερομηνίες) αποτελεί τον ακρογωνιαίο λίθο για μια επιτυχημένη εμπειρία χρήστη.

2.5 Μελέτες Περίπτωσης (Case Studies)

Η ανάλυση υφιστάμενων εφαρμογών από τον χώρο του θεάτρου παρέχει πολύτιμα συμπεράσματα σχετικά με τις τρέχουσες τάσεις και τις προσδοκίες του κοινού.

- **NT Live (National Theatre, UK):** Η εφαρμογή του Εθνικού Θεάτρου της Βρετανίας αποτελεί πρωτοπόρο παράδειγμα ψηφιακής διανομής θεατρικού περιεχομένου. Πέρα από την παροχή πληροφοριών, η κύρια λειτουργία της είναι η μετάδοση παραστάσεων μέσω streaming, επιτρέποντας σε ένα παγκόσμιο κοινό να απολαύσει υψηλής ποιότητας παραγωγές. Το παράδειγμα αυτό καταδεικνύει τη δυναμική των εφαρμογών να διευρύνουν το κοινό του θεάτρου πέρα από τα γεωγραφικά όρια του φυσικού χώρου.
- **Broadway Direct (USA):** Αυτή η εφαρμογή λειτουργεί ως μια κεντρική πύλη για το σύνολο των παραστάσεων που ανεβαίνουν στο Broadway της Νέας Υόρκης. Συγκεντρώνει πληροφορίες, ειδήσεις, προγράμματα και παρέχει απευθείας συνδέσμους για την αγορά εισιτηρίων. Η σημασία της έγκειται στο μοντέλο της συγκέντρωσης (aggregation), προσφέροντας στον χρήστη ένα μοναδικό σημείο αναφοράς (one-stop-shop) για ένα ολόκληρο θεατρικό οικοσύστημα.
- **Εθνικό Θέατρο Ελλάδος:** Η εφαρμογή του πρώτου κρατικού θεάτρου της χώρας αποτελεί ένα παράδειγμα απευθείας διαύλου επικοινωνίας ενός μεγάλου οργανισμού με το κοινό του. Συνδυάζει την παροχή πληροφοριών για το ρεπερτόριο, την ηλεκτρονική αγορά εισιτηρίων και την προβολή των εκπαιδευτικών του προγραμμάτων, διαχειριζόμενη ολιστικά την ψηφιακή σχέση με τους θεατές.

Η επισκόπηση αυτών των παραδειγμάτων επιβεβαιώνει ότι οι επιτυχημένες θεατρικές εφαρμογές λειτουργούν ως ολοκληρωμένες πολιτιστικές πύλες. Βελτιώνουν την εμπειρία του υφιστάμενου κοινού και, ταυτόχρονα, αποτελούν ένα ισχυρό εργαλείο για την προσέλκυση νέων θεατών, καθιστώντας το θεατρικό περιεχόμενο πιο προσιτό και άμεσο [1], [5].

2.6 Συμπεράσματα Θεωρητικού Υποβάθρου

Η ανάλυση που προηγήθηκε στο παρόν κεφάλαιο καταδεικνύει ότι η ανάπτυξη εφαρμογών για τον πολιτιστικό τομέα είναι ένας σύνθετος αλλά εξαιρετικά γόνιμος τομέας. Η εξέλιξη των τεχνολογιών και οι μεταβαλλόμενες συνήθειες του κοινού έχουν διαμορφώσει ένα περιβάλλον ώριμο για καινοτόμες ψηφιακές λύσεις.

Η επιλογή μιας cross-platform τεχνολογίας όπως το Flutter αποδεικνύεται στρατηγικά ορθή, καθώς επιτρέπει την ταχεία ανάπτυξη μιας εφαρμογής υψηλής απόδοσης και αισθητικής, η οποία μπορεί να απευθυνθεί στο ευρύτερο δυνατό κοινό (iOS και Android) με βελτιστοποιημένη διαχείριση πόρων [2], [6].

Παράλληλα, αναδείχθηκε η κρισιμότητα της εστίασης στη σχεδίαση της εμπειρίας χρήστη (UX) και της διεπαφής (UI). Σε έναν τομέα όπου η αισθητική και η αλληλεπίδραση είναι εξίσου σημαντικές με την πληροφορία, η προσεκτική σχεδίαση αποτελεί τον ακρογωνιαίο λίθο για τη δημιουργία μιας εφαρμογής που θα είναι όχι μόνο λειτουργική, αλλά και ελκυστική.

Τέλος, οι μελέτες περίπτωσης επιβεβαιώνουν την ύπαρξη μιας σαφούς ζήτησης και ανάγκης για κεντρικοποιημένες, καλοσχεδιασμένες πολιτιστικές πλατφόρμες. Το θεωρητικό υπόβαθρο που αναπτύχθηκε παρέχει, συνεπώς, μια στέρεη θεμελίωση για τη

σκοπιμότητα και τη δομή του έργου, οριοθετώντας το πλαίσιο για τα επόμενα κεφάλαια της ανάλυσης, του σχεδιασμού και της υλοποίησης.

3. Ανάλυση Απαιτήσεων

Η ανάλυση απαιτήσεων είναι η συστηματική διαδικασία προσδιορισμού, τεκμηρίωσης και διαχείρισης των αναγκών και των περιορισμών για ένα πληροφοριακό σύστημα. Στόχος της είναι να θέσει τις βάσεις για τον σχεδιασμό, ορίζοντας με ακρίβεια τις λειτουργίες που θα εκτελεί το σύστημα, τα ποιοτικά χαρακτηριστικά που θα το διέπουν και τους στόχους που καλείται να επιτύχει. Στην περίπτωση της εφαρμογής για το «Θεατρικό», η σχολαστική ανάλυση των απαιτήσεων διασφάλισε ότι η πολυπλοκότητα του θεατρικού περιεχομένου θα αποδοθεί με τρόπο λειτουργικό και κατανοητό, ευθυγραμμίζοντας τις τεχνικές προδιαγραφές με τις προσδοκίες των τελικών χρηστών.
Η διαδικασία που ακολουθήθηκε για το παρόν έργο περιελάμβανε τα εξής στάδια:

- **Καταγραφή των Λειτουργικών Απαιτήσεων (Functional Requirements):** Προσδιορισμός των συγκεκριμένων ενεργειών και λειτουργιών που πρέπει να μπορεί να εκτελέσει ο χρήστης μέσω της εφαρμογής.
- **Αποτύπωση των Μη Λειτουργικών Απαιτήσεων (Non-Functional Requirements):** Καθορισμός των ποιοτικών χαρακτηριστικών του συστήματος, όπως η απόδοση, η ασφάλεια και η ευχρηστία.
- **Προσδιορισμός των Χρηστών και των Σεναρίων Χρήσης (Use Cases):** Ανάλυση των διαφορετικών ομάδων χρηστών και περιγραφή των τυπικών αλληλεπιδράσεών τους με την εφαρμογή.
- **Δημιουργία Πίνακα Απαιτήσεων:** Συγκέντρωση και ιεράρχηση όλων των απαιτήσεων σε έναν ενοποιημένο πίνακα για την παρακολούθηση της υλοποίησής τους.

3.1 Λειτουργικές Απαιτήσεις (Functional Requirements)

Οι λειτουργικές απαιτήσεις περιγράφουν τις συγκεκριμένες συμπεριφορές και λειτουργίες του συστήματος. Ορίζουν «τι κάνει» η εφαρμογή. Για το παρόν έργο, οι βασικές λειτουργικές απαιτήσεις κατηγοριοποιούνται ως εξής:

3.1.1 Ανάκτηση και Παρουσίαση Δεδομένων

Η κύρια λειτουργία της εφαρμογής είναι να λειτουργεί ως ένας κεντρικός κατάλογος θεατρικών πληροφοριών.

- **Σύνδεση με το Backend API:** Η εφαρμογή πρέπει να επικοινωνεί αδιάλειπτα με το κεντρικό RESTful API (υλοποιημένο σε ASP.NET Core) για την ανάκτηση και αποστολή δεδομένων σε πραγματικό χρόνο.
- **Παρουσίαση Περιεχομένου:** Η εφαρμογή πρέπει να ανακτά και να παρουσιάζει με σαφή και οπτικά ελκυστικό τρόπο τις βασικές οντότητες του συστήματος: θεατρικές παραστάσεις (Productions), εκδηλώσεις-ημερομηνίες (Events), συντελεστές (Persons), και χώρους (Venues).
- **Αναζήτηση:** Πρέπει να παρέχεται λειτουργία αναζήτησης με λέξεις-κλειδιά, επιτρέποντας στους χρήστες να εντοπίζουν γρήγορα παραστάσεις, ηθοποιούς ή θέατρα.

- **Φιλτράρισμα και Ταξινόμηση:** Οι χρήστες πρέπει να μπορούν να φιλτράρουν τον κατάλογο των παραστάσεων με βάση κριτήρια όπως το είδος (π.χ. Δράμα, Κωμωδία) και η ημερομηνία, καθώς και να ταξινομούν τα αποτελέσματα.

3.1.2 Διαχείριση Προφίλ Χρήστη

Το σύστημα πρέπει να παρέχει στους εγγεγραμμένους χρήστες έναν προσωπικό χώρο για τη διαχείριση του λογαριασμού τους.

- **Αυθεντικοποίηση:** Οι χρήστες πρέπει να μπορούν να δημιουργήσουν λογαριασμό (Sign Up) και να συνδεθούν (Login).
- **Επεξεργασία Προφίλ:** Οι χρήστες πρέπει να έχουν τη δυνατότητα να επεξεργάζονται το όνομα χρήστη (username) τους και να προσθέτουν, να επεξεργάζονται ή να διαγράφουν συνδέσμους προς τα προφίλ τους στα κοινωνικά δίκτυα (Facebook, Instagram, YouTube).
- **Διαχείριση Πολυμέσων:** Οι χρήστες πρέπει να μπορούν να ανεβάζουν εικόνες στο προφίλ τους, είτε από τη συσκευή τους είτε μέσω URL, και να ορίζουν μία από αυτές ως φωτογραφία προφίλ.
- **Διαχείριση Ασφάλειας:** Οι χρήστες πρέπει να μπορούν να προσθέτουν και να επαληθεύουν τον αριθμό του κινητού τους τηλεφώνου, καθώς και να ενεργοποιούν ή να απενεργοποιούν την επαλήθευση δύο παραγόντων (2FA) για αυξημένη ασφάλεια.

3.1.3 Σύστημα Διεκδίκησης (Claiming System)

Μια κεντρική απαίτηση του έργου είναι η δυνατότητα επαλήθευσης και ανάθεσης περιεχομένου σε επαγγελματίες.

- **Διεκδίκησης Προφίλ Συντελεστή:** Οι χρήστες πρέπει να μπορούν να υποβάλουν αίτημα για να «διεκδικήσουν» το επίσημο προφίλ ενός συντελεστή (π.χ. ηθοποιού), επισυνάπτοντας αποδεικτικό ταυτοποίησης.
- **Διεκδίκησης Διαχείρισης Εκδήλωσης:** Πιστοποιημένοι χρήστες (π.χ. παραγωγοί) πρέπει να μπορούν να διεκδικήσουν τη διαχείριση της σελίδας μιας παράστασης, αποκτώντας δικαιώματα επεξεργασίας του περιεχομένου της.
- **Διαχείριση Αιτημάτων:** Οι διαχειριστές με τον ρόλο «Claims Manager» πρέπει να έχουν πρόσβαση σε μια λίστα με τα εκκρεμή αιτήματα και να μπορούν να τα εγκρίνουν ή να τα απορρίπτουν.

3.1.4 Αλληλεπίδραση και Εξατομίκευση

Για την ενίσχυση της εμπειρίας χρήστη, απαιτούνται οι παρακάτω λειτουργίες:

- **Push Notifications:** Η εφαρμογή πρέπει να μπορεί να αποστέλλει ειδοποιήσεις στους χρήστες για σημαντικές ενημερώσεις, όπως η προσθήκη μιας νέας παράστασης από έναν αγαπημένο τους σκηνοθέτη ή αλλαγές στο πρόγραμμα.
- **Αγαπημένα (Favorites/Wishlist):** Οι χρήστες πρέπει να μπορούν να προσθέτουν παραστάσεις σε μια προσωπική λίστα «αγαπημένων» για γρήγορη πρόσβαση και παρακολούθηση.

- **Κοινωνική Κοινοποίηση (Social Sharing):** Πρέπει να παρέχεται η δυνατότητα κοινοποίησης των λεπτομερειών μιας παράστασης σε εξωτερικές εφαρμογές και κοινωνικά δίκτυα.
- **Ενσωμάτωση Ημερολογίου (Calendar Integration):** Οι χρήστες πρέπει να μπορούν να προσθέτουν την ημερομηνία και ώρα μιας παράστασης απευθείας στο ημερολόγιο της συσκευής τους.

3.2 Μη Λειτουργικές Απαιτήσεις (Non-Functional Requirements)

Οι μη λειτουργικές απαιτήσεις ορίζουν τα ποιοτικά χαρακτηριστικά και τους περιορισμούς του συστήματος. Περιγράφουν «πώς» το σύστημα εκτελεί τις λειτουργίες του.

3.2.1 Απόδοση (Performance)

Χρόνος Απόκρισης: Η εφαρμογή πρέπει να έχει γρήγορους χρόνους απόκρισης, με τις περισσότερες αλληλεπιδράσεις του χρήστη και τις κλήσεις στο API να ολοκληρώνονται σε λιγότερο από 1-2 δευτερόλεπτα.
Caching: Το σύστημα πρέπει να χρησιμοποιεί μηχανισμούς προσωρινής αποθήκευσης (caching) τόσο στην πλευρά του εξυπηρετητή (server-side, με χρήση MemoryCache) όσο και στην πλευρά του πελάτη (client-side) για τη μείωση των επαναλαμβανόμενων κλήσεων στο API και τη βελτίωση της ταχύτητας φόρτωσης.
Κλιμακωσιμότητα (Scalability): Η αρχιτεκτονική του backend πρέπει να είναι σχεδιασμένη ώστε να μπορεί να εξυπηρετήσει έναν αυξανόμενο αριθμό ταυτόχρονων χρηστών χωρίς σημαντική υποβάθμιση της απόδοσης [12].

3.2.2 Ασφάλεια (Security)

Προστασία Δεδομένων: Το σύστημα πρέπει να διασφαλίζει την προστασία των προσωπικών δεδομένων των χρηστών, σύμφωνα με τις αρχές του Γενικού Κανονισμού για την Προστασία Δεδομένων (GDPR) [13].
Ασφαλής Επικοινωνία: Όλη η επικοινωνία μεταξύ της εφαρμογής-πελάτη και του API πρέπει να γίνεται μέσω κρυπτογραφημένου καναλιού (HTTPS) [13].
Εξουσιοδότηση: Η πρόσβαση σε λειτουργίες και δεδομένα του API πρέπει να ελέγχεται αυστηρά μέσω μηχανισμών εξουσιοδότησης, όπως τα JSON Web Tokens (JWT) [14], [13].

3.2.3 Ευχρηστία (Usability)

Διαισθητική Πλοήγηση: Το περιβάλλον χρήστη (UI) πρέπει να είναι καθαρό, συνεπές και διαισθητικό, επιτρέποντας στους χρήστες να επιτυγχάνουν τους στόχους τους με ελάχιστη προσπάθεια [4].
Προσβασιμότητα (Accessibility): Η εφαρμογή πρέπει να σχεδιαστεί λαμβάνοντας υπόψη τις βασικές αρχές προσβασιμότητας, ώστε να είναι χρησιμοποιήσιμη και από άτομα με αναπηρίες (π.χ., συμβατότητα με screen readers) [8], [11].

3.2.4 Επεκτασιμότητα και Συντηρησιμότητα (Extensibility & Maintainability)

Modular Αρχιτεκτονική: Τόσο το backend όσο και το frontend πρέπει να είναι δομημένα με μια αρθρωτή (modular) αρχιτεκτονική (π.χ., διαχωρισμός σε Services, Repositories, Controllers στο backend) που να επιτρέπει την εύκολη προσθήκη νέων λειτουργιών και τη συντήρηση του υπάρχοντος κώδικα.

3.2.5 Συμβατότητα (Compatibility)

Cross-Platform: Η επιλογή του Flutter ως τεχνολογίας ανάπτυξης διασφαλίζει ότι η εφαρμογή μπορεί να λειτουργήσει τόσο σε περιβάλλον iOS όσο και Android από μία ενιαία βάση κώδικα.
Συμβατότητα Λειτουργικού: Η εφαρμογή πρέπει να είναι συμβατή με τις πρόσφατες εκδόσεις των λειτουργικών συστημάτων iOS και Android.

3.3 Χρήστες και Σενάρια Χρήσης

Για τον σχεδιασμό του συστήματος, αναγνωρίστηκαν τρεις κύριες ομάδες χρηστών (user roles):

1. **Θεατής (End-User):** Η κύρια ομάδα-στόχος. Είναι ο απλός χρήστης που αναζητά πληροφορίες για θεατρικές παραστάσεις, θέλει να δει προγράμματα, συντελεστές και να οργανώσει την ψυχαγωγία του.
2. **Επαγγελματίας του Χώρου (Content Manager):** Περιλαμβάνει ηθοποιούς, σκηνοθέτες, παραγωγούς που, μετά από διαδικασία ταυτοποίησης (claim), αποκτούν δικαιώματα διαχείρισης του περιεχομένου που τους αφορά (π.χ. βιογραφικό, φωτογραφίες, λεπτομέρειες παράστασης).
3. **Διαχειριστής Συστήματος (Administrator):** Έχει πλήρη πρόσβαση στο σύστημα. Οι διαχειριστές χωρίζονται σε υπο-ρόλους, όπως ο «Claims Manager» που είναι υπεύθυνος για την έγκριση ή απόρριψη των αιτημάτων διεκδίκησης.

Βασικά Σενάρια Χρήσης (Use Cases)

- **UC-01: Αναζήτηση και Προβολή Παράστασης (Θεατής)**
    1. Ο χρήστης ανοίγει την εφαρμογή και βλέπει την αρχική οθόνη με μια λίστα παραστάσεων.
    2. Χρησιμοποιεί τη γραμμή αναζήτησης για να βρει μια συγκεκριμένη παράσταση.
    3. Επιλέγει την παράσταση από τα αποτελέσματα.
    4. Η εφαρμογή εμφανίζει μια οθόνη με όλες τις λεπτομέρειες: σύνοψη, φωτογραφίες, λίστα συντελεστών, ημερομηνίες, θέατρο και τιμές εισιτηρίων.
- **UC-02: Επεξεργασία Προφίλ (Θεατής/Επαγγελματίας)**
    1. Ο χρήστης πλοηγείται στην οθόνη του προφίλ του.
    2. Επιλέγει την επιλογή «Επεξεργασία».
    3. Προσθέτει τον σύνδεσμο του προφίλ του στο Instagram.
    4. Πατάει «Αποθήκευση».

5. Το σύστημα ενημερώνει τα στοιχεία του χρήστη στη βάση δεδομένων και η αλλαγή εμφανίζεται στο προφίλ του.

- **UC-03: Διεκδίκηση Προφίλ Ηθοποιού (Επαγγελματίας)**
    1. Ο χρήστης (που είναι ηθοποιός) βρίσκει το προφίλ του στην εφαρμογή.
    2. Πατάει το κουμπί «Claim Profile».
    3. Η εφαρμογή τον καθοδηγεί να ανεβάσει ένα έγγραφο ταυτοποίησης.
    4. Το σύστημα δημιουργεί ένα αίτημα (Account Request) και ειδοποιεί τους Claims Managers.
    5. Μετά την έγκριση, ο λογαριασμός του χρήστη συνδέεται με το προφίλ του ηθοποιού.

3.4 Πίνακας Απαιτήσεων

Για την καλύτερη οργάνωση και παρακολούθηση, οι απαιτήσεις συγκεντρώθηκαν στον παρακάτω πίνακα:

Πίνακας 3.1: Συγκεντρωτικός πίνακας λειτουργικών και μη λειτουργικών απαιτήσεων

ID	Περιγραφή Απαίτησης	Τύπος	Προτεραιότητα
FR-01	Ανάκτηση και προβολή λίστας παραστάσεων	Λειτουργική	Υψηλή
FR-02	Προβολή λεπτομερειών παράστασης (συντελεστές, ημερομηνίες, κ.λπ.)	Λειτουργική	Υψηλή
FR-03	Σύστημα εγγραφής και σύνδεσης χρηστών	Λειτουργική	Υψηλή
FR-04	Επεξεργασία προφίλ χρήστη (username, social media)	Λειτουργική	Υψηλή
FR-05	Δυνατότητα χρήστη να ανεβάζει και να διαχειρίζεται φωτογραφίες	Λειτουργική	Μεσαία
FR-06	Σύστημα διεκδίκησης (claim) προφίλ συντελεστών και παραστάσεων	Λειτουργική	Υψηλή
FR-07	Σύστημα διαχείρισης αιτημάτων claim από διαχειριστές	Λειτουργική	Υψηλή
FR-08	Ενεργοποίηση/Απενεργοποίηση 2-Factor Authentication (2FA)	Λειτουργική	Μεσαία
FR-09	Λειτουργία προσθήκης παράστασης στα "Αγαπημένα"	Λειτουργική	Μεσαία
FR-10	Αποστολή Push Notifications	Λειτουργική	Χαμηλή
NFR-01	Χρόνος απόκρισης API < 500ms για τις περισσότερες κλήσεις	Μη Λειτουργική	Υψηλή

NFR-02	Χρήση caching για τη βελτίωση της απόδοσης	Μη Λειτουργική	Υψηλή
NFR-03	Κρυπτογραφημένη επικοινωνία (HTTPS)	Μη Λειτουργική	Υψηλή
NFR-04	Έλεγχος πρόσβασης μέσω JWT	Μη Λειτουργική	Υψηλή
NFR-05	Διαισθητικό και συνεπές UI	Μη Λειτουργική	Υψηλή
NFR-06	Αρθρωτή αρχιτεκτονική κώδικα για εύκολη επεκτασιμότητα	Μη Λειτουργική	Υψηλή
NFR-07	Cross-platform συμβατότητα (iOS/Android)	Μη Λειτουργική	Υψηλή

3.5 Συμπεράσματα Ανάλυσης Απαιτήσεων

Η ολοκλήρωση της φάσης ανάλυσης απαιτήσεων παρήγαγε ένα σαφές και ολοκληρωμένο προσχέδιο για την εφαρμογή «Θεατρικό». Μέσω της διαδικασίας αυτής, ορίστηκαν με ακρίβεια οι βασικές λειτουργίες, όπως η παρουσίαση του θεατρικού περιεχομένου, η διαχείριση των χρηστών και το καινοτόμο σύστημα διεκδίκησης, οι οποίες αποτελούν τον πυρήνα του συστήματος. Ταυτόχρονα, οι μη λειτουργικές απαιτήσεις έθεσαν τα ποιοτικά πρότυπα για την απόδοση, την ασφάλεια και την ευχρηστία, διασφαλίζοντας ότι το τελικό προϊόν θα είναι αξιόπιστο και ποιοτικό. Η σαφής αυτή καταγραφή λειτούργησε ως οδικός χάρτης για τα επόμενα στάδια του σχεδιασμού και της υλοποίησης, εγγυώμενη ότι η ανάπτυξη θα παραμείνει εστιασμένη στις πραγματικές ανάγκες των χρηστών και στους αρχικούς στόχους του έργου.

4. Σχεδίαση Συστήματος

Μετά την ολοκλήρωση της ανάλυσης των απαιτήσεων, το επόμενο κρίσιμο στάδιο στην ανάπτυξη ενός πληροφοριακού συστήματος είναι η σχεδίαση. Σε αυτή τη φάση, οι αφηρημένες απαιτήσεις μετατρέπονται σε συγκεκριμένες αρχιτεκτονικές αποφάσεις, δομές δεδομένων και τεχνολογικές επιλογές που θα καθορίσουν τον τρόπο λειτουργίας του συστήματος. Το παρόν κεφάλαιο παρουσιάζει τη σχεδίαση της εφαρμογής «Θεατρικού», αναλύοντας την αρχιτεκτονική του, τον τρόπο επικοινωνίας των επιμέρους συστατικών του, τη διαχείριση των δεδομένων, τις αρχές σχεδίασης της διεπαφής χρήστη, καθώς και τις στρατηγικές που υιοθετήθηκαν για την ασφάλεια, την επεκτασιμότητα και τη συντηρησιμότητα του συστήματος. Η λεπτομερής αυτή παρουσίαση θα συνοδευτεί από διαγράμματα που οπτικοποιούν τις δομές και τις αλληλεπιδράσεις.

4.1 Αρχιτεκτονική Συστήματος

Η αρχιτεκτονική του «Θεατρικού» βασίζεται σε ένα σύγχρονο **μοντέλο Client-Server**, το οποίο διαχωρίζει σαφώς τις αρμοδιότητες μεταξύ της εφαρμογής-πελάτη (frontend) και της υποστηρικτικής υποδομής (backend). Αυτή η διάταξη επιτρέπει την ανεξάρτητη ανάπτυξη, κλιμάκωση και συντήρηση των δύο μερών, προσφέροντας παράλληλα ευελιξία και ανθεκτικότητα.

Συγκεκριμένα, το σύστημα αποτελείται από:

1. **Εφαρμογή Πελάτη (Client Application):** Πρόκειται για την εφαρμογή για κινητές συσκευές, η οποία αναπτύχθηκε με το **Flutter framework** και τη γλώσσα **Dart**. Η εφαρμογή αυτή είναι υπεύθυνη για την παρουσίαση του γραφικού περιβάλλοντος χρήστη (UI), την αλληλεπίδραση με τον χρήστη και την αποστολή αιτημάτων προς το backend API. Η επιλογή του Flutter διασφαλίζει την cross-platform συμβατότητα, επιτρέποντας τη λειτουργία της εφαρμογής τόσο σε περιβάλλον iOS όσο και Android από μία ενιαία βάση κώδικα.
2. **Υποδομή Εξυπηρετητή (Backend Infrastructure):** Αποτελείται από ένα **RESTful API**, υλοποιημένο σε **ASP.NET Core** με τη γλώσσα **C#**. Το backend είναι υπεύθυνο για τη διαχείριση της επιχειρησιακής λογικής, την αυθεντικοποίηση και εξουσιοδότηση των χρηστών, την επικοινωνία με τη βάση δεδομένων και την ενσωμάτωση με εξωτερικές υπηρεσίες.
3. **Βάση Δεδομένων (Database):** Χρησιμοποιείται μια σχεσιακή βάση δεδομένων **PostgreSQL**, η οποία φιλοξενεί όλα τα δομημένα δεδομένα του συστήματος, όπως πληροφορίες για παραστάσεις, συντελεστές, θέατρα, χρήστες και τις μεταξύ τους σχέσεις [15].
4. **Υπηρεσία Αποθήκευσης Αντικειμένων (Object Storage Service):** Για την αποθήκευση μη δομημένων δεδομένων, όπως εικόνες προφίλ χρηστών ή αφίσες παραστάσεων, χρησιμοποιείται η υπηρεσία **Minio**.

Η επικοινωνία μεταξύ της εφαρμογής-πελάτη και του backend API πραγματοποιείται μέσω του πρωτοκόλλου HTTP/HTTPS, με ανταλλαγή δεδομένων σε μορφή JSON. Αυτή η αρχιτεκτονική επιτρέπει την εύκολη κλιμάκωση κάθε επιμέρους συστατικού και την ανεξάρτητη εξέλιξή του.

4.2 Το API και η Επικοινωνία με τη Βάση Δεδομένων

Το RESTful API, υλοποιημένο σε ASP.NET Core, αποτελεί τον κεντρικό κόμβο της αρχιτεκτονικής, διαχειριζόμενο όλες τις αιτήσεις από την εφαρμογή-πελάτη και αλληλεπιδρώντας με τη βάση δεδομένων. Η σχεδίαση του API ακολουθεί τις αρχές του **Clean Architecture** και του **Domain-Driven Design**, διαχωρίζοντας σαφώς τις αρμοδιότητες και προωθώντας την επαναχρησιμοποίηση του κώδικα [16].

- **Controllers:** Αποτελούν το σημείο εισόδου για τα αιτήματα HTTP. Είναι υπεύθυνα για την παραλαβή των αιτημάτων, την επικύρωση των εισερχόμενων δεδομένων και την ανάθεση της επιχειρησιακής λογικής στις υπηρεσίες (Services) [17].
- **Services:** Περιέχουν την κύρια επιχειρησιακή λογική της εφαρμογής. Για παράδειγμα, η IUserService (όπως φαίνεται στο Program.cs) είναι υπεύθυνη για λειτουργίες που αφορούν τους χρήστες, όπως η ανάκτηση προφίλ, η ενημέρωση κοινωνικών δικτύων ή η διαχείριση του 2FA.
- **Repositories:** Υλοποιούν το **Repository Pattern**, το οποίο αφαιρεί την πολυπλοκότητα της πρόσβασης στα δεδομένα από την επιχειρησιακή λογική. Κάθε IRepository (π.χ., IPersonRepository, IUserRepository) παρέχει μεθόδους για την εκτέλεση CRUD (Create, Read, Update, Delete) λειτουργιών σε συγκεκριμένες οντότητες της βάσης δεδομένων [12].
- **Entity Framework Core (EF Core):** Χρησιμοποιείται ως Object-Relational Mapper (ORM) για την επικοινωνία με τη βάση δεδομένων PostgreSQL. Το EF Core επιτρέπει την αλληλεπίδραση με τη βάση δεδομένων χρησιμοποιώντας αντικείμενα C# (entities) αντί για απευθείας SQL queries, απλοποιώντας την ανάπτυξη και μειώνοντας τον κίνδυνο σφαλμάτων. Η σύνδεση με τη βάση δεδομένων ορίζεται στο Program.cs μέσω του builder.Services.AddDbContext<TheatricalPlaysDbContext>(opt => opt.UseNpgsql(...)) [18].
- **Validation Services:** Εξειδικευμένες υπηρεσίες (π.χ., IUserValidationService) είναι υπεύθυνες για την επικύρωση των δεδομένων πριν αυτά αποθηκευτούν ή επεξεργαστούν, διασφαλίζοντας την ακεραιότητα και την ορθότητα των πληροφοριών.

Η επικοινωνία του API με τη βάση δεδομένων είναι διαφανής για την εφαρμογή-πελάτη. Η Flutter εφαρμογή στέλνει αιτήματα HTTP στο API, το API επεξεργάζεται τα αιτήματα, αλληλεπιδρά με τη βάση δεδομένων μέσω του EF Core και επιστρέφει την απάντηση στην εφαρμογή-πελάτη.

4.3 Διαχείριση Δεδομένων

Η αποτελεσματική διαχείριση των δεδομένων είναι θεμελιώδης για τη λειτουργικότητα και την απόδοση του συστήματος. Το «Θεατρικό» διαχειρίζεται δύο κύριους τύπους δεδομένων:

1. **Δομημένα Δεδομένα:** Αποθηκεύονται στη σχεσιακή βάση δεδομένων **PostgreSQL**. Το σχήμα της βάσης δεδομένων, όπως προκύπτει από το αρχείο seed.sql και τον κώδικα στο Program.cs, περιλαμβάνει πίνακες για:
    - system: Πληροφορίες για το ίδιο το σύστημα.
    - roles: Κατηγορίες ρόλων (π.χ., Ηθοποιός, Σκηνοθέτης).

- authorities: Ρόλοι χρηστών (π.χ., admin, user, claims manager).
- organizer: Διοργανωτές παραστάσεων.
- venue: Θεατρικοί χώροι.
- persons: Συντελεστές (ηθοποιοί, σκηνοθέτες κ.λπ.).
- production: Θεατρικές παραστάσεις.
- events: Συγκεκριμένες ημερομηνίες και ώρες παραστάσεων.
- contributions: Συσχετίζει συντελεστές με παραστάσεις και ρόλους.
- users: Εγγεγραμμένοι χρήστες της εφαρμογής.
- userimages: Εικόνες που ανεβάζουν οι χρήστες.
- accountrequests: Αιτήματα διεκδίκησης προφίλ.

Η αρχική συμπλήρωση της βάσης δεδομένων με δεδομένα (data seeding) πραγματοποιείται αυτόματα κατά την πρώτη εκκίνηση της εφαρμογής, όπως φαίνεται στο Program.cs (if (!db.Systems.Any())). Αυτό διασφαλίζει ότι το σύστημα είναι άμεσα λειτουργικό με ένα βασικό σύνολο πληροφοριών.

2. **Μη Δομημένα Δεδομένα (Object Storage):** Για την αποθήκευση αρχείων όπως εικόνες προφίλ χρηστών ή άλλων πολυμέσων, χρησιμοποιείται η υπηρεσία **Minio**. Το Minio είναι ένα open-source object storage server συμβατό με το Amazon S3 API. Αυτή η προσέγγιση είναι ιδανική για μεγάλο όγκο αρχείων, καθώς προσφέρει υψηλή διαθεσιμότητα, επεκτασιμότητα και απόδοση, χωρίς να επιβαρύνει τη σχεσιακή βάση δεδομένων. Η ενσωμάτωση του Minio γίνεται μέσω του IMinioService στο backend, όπως φαίνεται στο Program.cs.

Επιπλέον, το σύστημα χρησιμοποιεί μηχανισμούς **caching** (μέσω IMemoryCache και ICaching στο backend) για την προσωρινή αποθήκευση συχνά προσπελάσιμων δεδομένων, μειώνοντας τον φόρτο στη βάση δεδομένων και βελτιώνοντας τους χρόνους απόκρισης.

4.4 Σχεδίαση UI/UX

Η σχεδίαση της διεπαφής χρήστη (UI) και της εμπειρίας χρήστη (UX) για την εφαρμογή Flutter ήταν κεντρικής σημασίας, με στόχο τη δημιουργία ενός ελκυστικού, διαισθητικού και λειτουργικού περιβάλλοντος. Οι αρχές που καθοδήγησαν τη σχεδίαση περιλαμβάνουν:

- **Απλότητα και Καθαρότητα:** Το UI σχεδιάστηκε ώστε να είναι λιτό και ευανάγνωστο, αποφεύγοντας την υπερφόρτωση πληροφοριών. Η πλοήγηση είναι άμεση και διαισθητική, επιτρέποντας στους χρήστες να βρίσκουν εύκολα αυτό που αναζητούν.
- **Συνέπεια:** Εφαρμόστηκε ένα συνεκτικό οπτικό στυλ σε όλη την εφαρμογή, χρησιμοποιώντας μια καθορισμένη παλέτα χρωμάτων (MyColors.dart), τυπογραφία και εικονίδια. Αυτό ενισχύει την αναγνωρισιμότητα και την αίσθηση οικειότητας.
- **Ευκολία Χρήσης (Usability):** Οι λειτουργίες είναι προσβάσιμες με ελάχιστα βήματα. Για παράδειγμα, η οθόνη EditProfileScreen.dart οργανώνει τις επιλογές επεξεργασίας προφίλ (κοινωνικά δίκτυα, username, 2FA, τηλέφωνο) με σαφήνεια, χρησιμοποιώντας TextEditingController για την εύκολη εισαγωγή

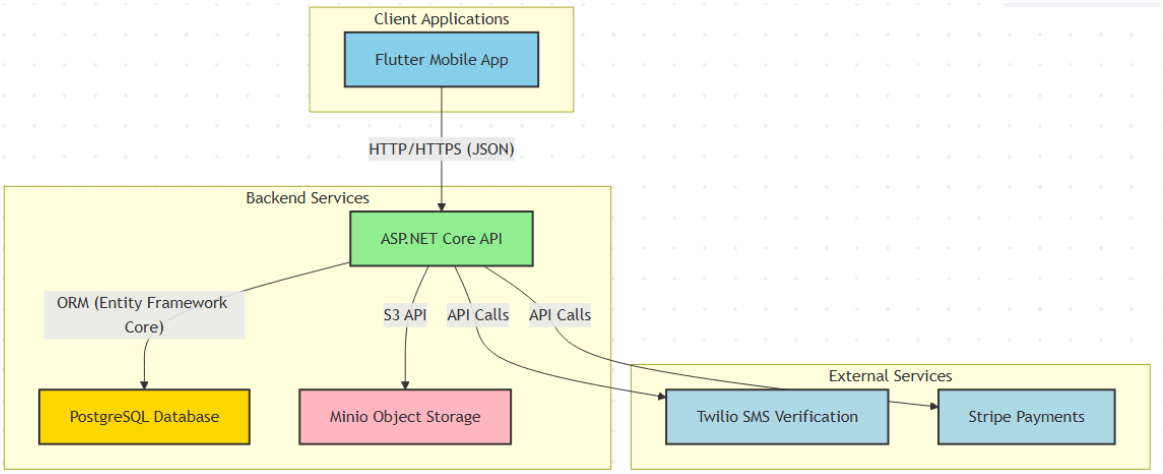
δεδομένων και CupertinoSwitch για την ενεργοποίηση/απενεργοποίηση λειτουργιών.

- **Προσαρμοστικότητα (Responsiveness):** Η σχεδίαση λαμβάνει υπόψη τις διαφορετικές διαστάσεις οθονών των κινητών συσκευών, διασφαλίζοντας ότι η εφαρμογή εμφανίζεται σωστά και είναι πλήρως λειτουργική σε κάθε συσκευή.
- **Οπτική Ανατροφοδότηση (Visual Feedback):** Η εφαρμογή παρέχει άμεση οπτική ανατροφοδότηση στον χρήστη για τις ενέργειές του, π.χ., μέσω εικονιδίων επιτυχίας/αποτυχίας (π.χ., πράσινο checkmark για επαληθευμένο τηλέφωνο στο EditProfileScreen.dart) και ειδοποιήσεων (AwesomeNotifications).
- **Dark/Light Mode:** Η εφαρμογή υποστηρίζει εναλλαγή μεταξύ Dark και Light Mode, μια λειτουργία που βελτιώνει την εμπειρία χρήστη και την προσβασιμότητα, ειδικά σε συνθήκες χαμηλού φωτισμού. Η προτίμηση αποθηκεύεται μέσω SharedPreferences και διαχειρίζεται από το MyApp (βλ. main.dart).

4.5 Διαγράμματα Συστήματος

Για την οπτικοποίηση της αρχιτεκτονικής και των αλληλεπιδράσεων του συστήματος, παρουσιάζονται τα ακόλουθα διαγράμματα:

Διάγραμμα Αρχιτεκτονικής Συστήματος (System Architecture Diagram)



Σχήμα 4.1: Διάγραμμα Αρχιτεκτονικής Συστήματος (System Architecture Diagram)

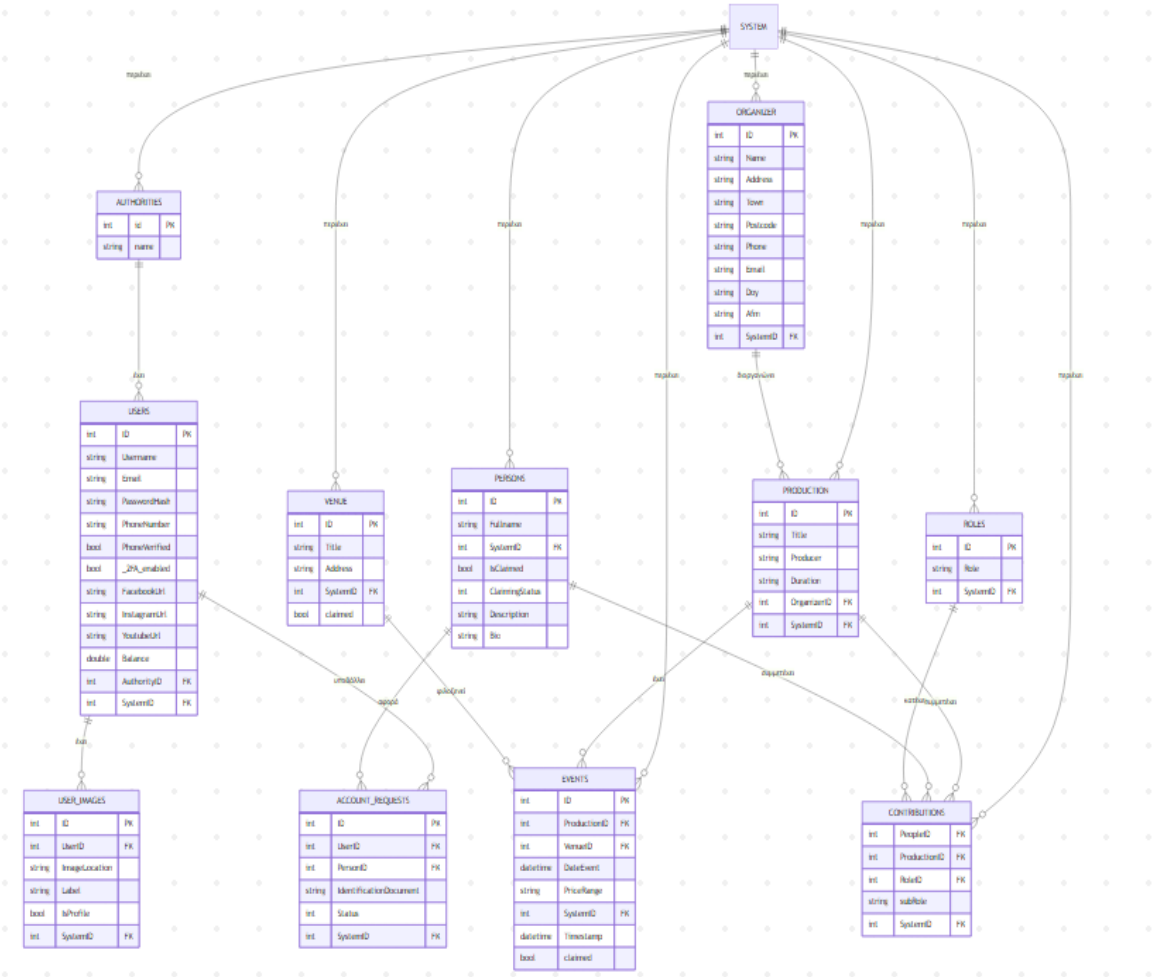
Το διάγραμμα αυτό παρέχει μια υψηλού επιπέδου, πανοραμική όψη ολόκληρου του πληροφοριακού συστήματος. Σκοπός του είναι να απεικονίσει τα θεμελιώδη δομικά στοιχεία της εφαρμογής και τον τρόπο με τον οποίο αυτά επικοινωνούν και συνεργάζονται για να παραδώσουν την τελική λειτουργικότητα στον χρήστη.

Ανάλυση Στοιχείων:

- **Client Applications (Flutter Mobile App):** Αντιπροσωπεύει το **frontend** του συστήματος, δηλαδή την εφαρμογή για κινητά με την οποία αλληλεπιδρά ο τελικός χρήστης. Είναι υπεύθυνο για την παρουσίαση του γραφικού περιβάλλοντος (UI) και την αποστολή αιτημάτων προς το backend.

- **Backend Services:** Αυτό είναι το "κέντρο ελέγχου" του συστήματος.
    - **ASP.NET Core API:** Είναι ο "εγκέφαλος" της εφαρμογής. Λαμβάνει τα αιτήματα από την εφαρμογή-πελάτη, εκτελεί την επιχειρησιακή λογική (π.χ. έλεγχος δικαιωμάτων, επεξεργασία δεδομένων) και συντονίζει την επικοινωνία με τα υπόλοιπα μέρη.
    - **PostgreSQL Database:** Είναι η σχεσιακή βάση δεδομένων όπου αποθηκεύονται όλα τα δομημένα δεδομένα, όπως οι χρήστες, οι παραστάσεις, οι συντελεστές και οι μεταξύ τους σχέσεις.
    - **Minio Object Storage:** Λειτουργεί ως αποθήκη για μη δομημένα δεδομένα, κυρίως αρχεία πολυμέσων όπως οι εικόνες που ανεβάζουν οι χρήστες. Η χρήση του αποφορτίζει τη βασική βάση δεδομένων και βελτιστοποιεί την απόδοση.
- **External Services:** Πρόκειται για υπηρεσίες τρίτων που το σύστημα χρησιμοποιεί για εξειδικευμένες λειτουργίες.
    - **Twilio SMS Verification:** Χρησιμοποιείται για την αποστολή κωδικών μιας χρήσης (OTP) μέσω SMS, για την επαλήθευση του αριθμού τηλεφώνου των χρηστών και την υποστήριξη του 2-Factor Authentication.
    - **Stripe Payments:** (Αν και δεν υλοποιήθηκε πλήρως στο παρόν project, προβλέπεται αρχιτεκτονικά) Θα διαχειριζόταν τις ασφαλείς ηλεκτρονικές πληρωμές, π.χ. για την αγορά credits.
- **Γραμμές Επικοινωνίας:** Οι γραμμές δείχνουν πώς "μιλούν" τα στοιχεία μεταξύ τους. Για παράδειγμα, η εφαρμογή Flutter επικοινωνεί με το API μέσω **HTTP/HTTPS**, ανταλλάσσοντας δεδομένα σε μορφή **JSON**. Το API, με τη σειρά του, επικοινωνεί με τη βάση PostgreSQL μέσω του **Entity Framework Core (ORM)**.

Διάγραμμα Σχέσεων Οντοτήτων (Entity-Relationship Diagram - ERD)



Σχήμα 4.2: Διάγραμμα Σχέσεων Οντοτήτων (Entity-Relationship Diagram)

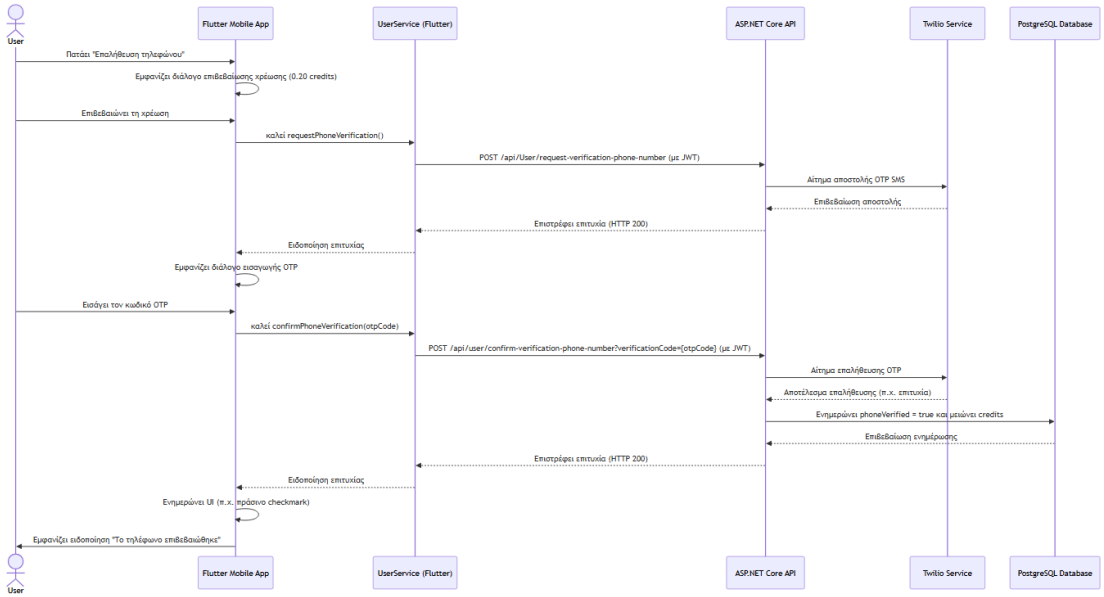
Το ERD είναι το "αρχιτεκτονικό σχέδιο" της βάσης δεδομένων PostgreSQL. Σκοπός του είναι να οπτικοποιήσει τις οντότητες (δηλαδή τους πίνακες) της βάσης, τα χαρακτηριστικά τους (τις στήλες) και, κυρίως, τις λογικές σχέσεις που τις συνδέουν.

Ανάλυση Στοιχείων:

- **Οντότητες (Entities):** Κάθε ορθογώνιο στο διάγραμμα αντιστοιχεί σε έναν πίνακα στη βάση δεδομένων. Για παράδειγμα, έχουμε τις οντότητες PRODUCTION (Παραστάσεις), PERSONS (Συντελεστές), VENUE (Θέατρα), USERS (Χρήστες) κ.λπ.
- **Χαρακτηριστικά (Attributes):** Μέσα σε κάθε οντότητα αναγράφονται τα βασικά της χαρακτηριστικά, που αντιστοιχούν στις στήλες του πίνακα (π.χ., ο πίνακας PERSONS έχει στήλες ID, Fullname, Bio κ.ά.). Τα PK (Primary Key) και FK (Foreign Key) υποδεικνύουν τα πρωτεύοντα και ξένα κλειδιά αντίστοιχα, που είναι θεμελιώδη για τη δημιουργία σχέσεων.
- **Σχέσεις (Relationships):** Οι γραμμές που συνδέουν τις οντότητες δείχνουν πώς αυτές συσχετίζονται. Η σήμανση στα άκρα των γραμμών (γνωστή ως "crow's foot notation") περιγράφει την πληθικότητα της σχέσης (ένα-προς-ένα, ένα-προς-πολλά, πολλά-προς-πολλά).

- **Παράδειγμα:** Η οντότητα CONTRIBUTIONS (Συμμετοχές) είναι ένας κλασικός **πίνακας-γέφυρα** που υλοποιεί μια σχέση **πολλά-προς-πολλά**. Μια PRODUCTION (παράσταση) μπορεί να έχει πολλούς PERSONS (συντελεστές), και ένας PERSON μπορεί να συμμετέχει σε πολλές PRODUCTIONS. Ο πίνακας CONTRIBUTIONS συνδέει αυτούς τους δύο, καταγράφοντας για κάθε συμμετοχή ποιος συντελεστής (PeopleID) συμμετείχε σε ποια παράσταση (ProductionID) και με ποιον ρόλο (RoleID).

Διάγραμμα Ακολουθίας (Sequence Diagram) για το σενάριο "Επαλήθευση Αριθμού Τηλεφώνου"



Σχήμα 4.3: Διάγραμμα Ακολουθίας για το σενάριο «Επαλήθευση Αριθμού Τηλεφώνου»

Σε αντίθεση με τα προηγούμενα διαγράμματα που δείχνουν τη στατική δομή, το Διάγραμμα Ακολουθίας είναι **δυναμικό**. Σκοπός του είναι να απεικονίσει τη χρονική αλληλουχία των μηνυμάτων και των αλληλεπιδράσεων μεταξύ των διαφόρων συστατικών του συστήματος κατά την εκτέλεση μιας συγκεκριμένης λειτουργίας ή σεναρίου χρήσης.

Ανάλυση Σεναρίου "Επαλήθευση Αριθμού Τηλεφώνου":

Το διάγραμμα αυτό "αφηγείται" βήμα-βήμα τι συμβαίνει "πίσω από τις κουρτίνες" όταν ο χρήστης προσπαθεί να επαληθεύσει το τηλέφωνό του:

1. **Έναρξη από τον Χρήστη:** Ο χρήστης (User) ξεκινά τη διαδικασία πατώντας το σχετικό κουμπί στην εφαρμογή (FlutterApp).
2. **Επικοινωνία Εντός της Εφαρμογής:** Η οθόνη της εφαρμογής καλεί την εσωτερική της υπηρεσία (UserService) για να διαχειριστεί τη λογική.

3. **Αίτημα στο Backend:** Η UserService στέλνει ένα αίτημα HTTP POST στο ASP.NET Core API για να ζητήσει την αποστολή κωδικού. Το αίτημα περιλαμβάνει το JWT token για την ταυτοποίηση του χρήστη.
4. **Αλληλεπίδραση με Εξωτερική Υπηρεσία:** Το API, με τη σειρά του, δεν στέλνει το SMS μόνο του. Καλεί την εξωτερική υπηρεσία Twilio, η οποία αναλαμβάνει την αποστολή του κωδικού OTP στον αριθμό του χρήστη.
5. **Εισαγωγή από τον Χρήστη:** Ο χρήστης λαμβάνει το SMS και εισάγει τον κωδικό στην εφαρμογή.
6. **Διαδικασία Επιβεβαίωσης:** Η εφαρμογή στέλνει ξανά ένα αίτημα στο API, αυτή τη φορά για να επιβεβαιώσει τον κωδικό που έδωσε ο χρήστης.
7. **Επικύρωση και Ενημέρωση Βάσης:** Το API επικοινωνεί ξανά με το Twilio για να επικυρώσει τον κωδικό. Αν είναι σωστός, το API δίνει εντολή στη βάση δεδομένων (PostgreSQL Database) να ενημερώσει την κατάσταση του χρήστη (π.χ., phoneVerified = true) και να μειώσει το υπόλοιπό του (credits).
8. **Ανατροφοδότηση στον Χρήστη:** Το σύστημα επιστρέφει μηνύματα επιτυχίας σε όλη την αλυσίδα (από τη Βάση στο API, από το API στην εφαρμογή), και τελικά η εφαρμογή εμφανίζει μια ειδοποίηση επιτυχίας στον χρήστη, ενημερώνοντας ταυτόχρονα και το γραφικό περιβάλλον.

4.6 Η Ασφάλεια και η Προστασία των Δεδομένων

Η ασφάλεια και η προστασία των δεδομένων αποτελούν θεμελιώδεις πυλώνες στη σχεδίαση του συστήματος, δεδομένης της διαχείρισης προσωπικών δεδομένων και της ανάγκης για διασφάλιση της ακεραιότητας του περιεχομένου.

- **Αυθεντικοποίηση (Authentication):** Το σύστημα χρησιμοποιεί **JSON Web Tokens (JWT)** για την ταυτοποίηση των χρηστών. Μετά την επιτυχή σύνδεση, το API εκδίδει ένα JWT, το οποίο η εφαρμογή-πελάτης αποθηκεύει (π.χ., στο globalAccessToken στο UserService.dart) και το επισυνάπτει σε κάθε επόμενο αίτημα προς το API. Η διαμόρφωση του JWT authentication γίνεται στο Program.cs του backend.
- **Εξουσιοδότηση (Authorization):** Η πρόσβαση σε συγκεκριμένες λειτουργίες και πόρους του API ελέγχεται αυστηρά μέσω **Authorization Filters** (π.χ., AdminAuthorizationFilter, ClaimsManagerAuthorizationFilter, UserA uthorizationFilter). Αυτοί οι φίλτροι, όπως δηλώνονται στο Program.cs, διασφαλίζουν ότι μόνο οι χρήστες με τα κατάλληλα δικαιώματα μπορούν να εκτελέσουν συγκεκριμένες ενέργειες.
- **Επαλήθευση Δύο Παραγόντων (Two-Factor Authentication - 2FA):** Για την ενίσχυση της ασφάλειας των λογαριασμών, οι χρήστες έχουν τη δυνατότητα να ενεργοποιήσουν το 2FA. Αυτό υλοποιείται με την ενσωμάτωση της υπηρεσίας **Twilio** για την αποστολή κωδικών μιας χρήσης (OTP) μέσω SMS στον επαληθευμένο αριθμό τηλεφώνου του χρήστη. Η λογική για την ενεργοποίηση/απενεργοποίηση και την επαλήθευση του τηλεφώνου βρίσκεται στο EditProfileScreen.dart και UserService.dart.
- **Ασφαλής Επικοινωνία (HTTPS):** Όλη η επικοινωνία μεταξύ της εφαρμογής-πελάτη και του backend API πραγματοποιείται μέσω του πρωτοκόλλου **HTTPS**, διασφαλίζοντας την κρυπτογράφηση των δεδομένων κατά τη μεταφορά και την προστασία από υποκλοπές.

- **Προστασία Προσωπικών Δεδομένων (GDPR Compliance):** Ο σχεδιασμός του συστήματος λαμβάνει υπόψη τις αρχές του Γενικού Κανονισμού για την Προστασία Δεδομένων (GDPR), διασφαλίζοντας τη συλλογή, αποθήκευση και επεξεργασία των προσωπικών δεδομένων με διαφάνεια και ασφάλεια.

4.7 Επεκτασιμότητα και Συντηρησιμότητα

Η σχεδίαση του συστήματος δόθηκε ιδιαίτερη έμφαση στην επεκτασιμότητα και τη συντηρησιμότητα, ώστε να μπορεί να ανταποκριθεί σε μελλοντικές ανάγκες και αλλαγές χωρίς να απαιτούνται εκτεταμένες αναθεωρήσεις.

- **Αρθρωτή (Modular) Αρχιτεκτονική:** Τόσο το frontend (Flutter) όσο και το backend (ASP.NET Core) είναι δομημένα σε διακριτές, ανεξάρτητες ενότητες (modules). Στο backend, αυτό επιτυγχάνεται μέσω του διαχωρισμού σε Repositories, Services και Controllers, καθώς και της χρήσης Dependency Injection (βλ. Program.cs για την καταχώριση των υπηρεσιών). Αυτή η δομή διευκολύνει την προσθήκη νέων λειτουργιών ή την τροποποίηση υφιστάμενων χωρίς να επηρεάζονται άλλα μέρη του συστήματος.
- **Χρήση Σύγχρονων Frameworks και Γλωσσών:** Η επιλογή του Flutter/Dart και του ASP.NET Core/C# παρέχει πρόσβαση σε ένα πλούσιο οικοσύστημα εργαλείων, βιβλιοθηκών και κοινοτήτων, διευκολύνοντας την ανάπτυξη και την επίλυση προβλημάτων.
- **Αποσύνδεση (Decoupling):** Η αρχιτεκτονική Client-Server και η χρήση RESTful API διασφαλίζουν την πλήρη αποσύνδεση του frontend από το backend. Αυτό σημαίνει ότι κάθε μέρος μπορεί να αναπτυχθεί, να δοκιμαστεί και να αναπτυχθεί ανεξάρτητα [16].
- **Διαχείριση Δεδομένων (Data Management):** Η χρήση PostgreSQL ως σχεσιακής βάσης δεδομένων και Minio για object storage προσφέρει επεκτασιμότητα στην αποθήκευση δεδομένων. Το EF Core απλοποιεί τις αλλαγές στο σχήμα της βάσης δεδομένων (migrations).
- **Curators:** Η ύπαρξη "Curator" υπηρεσιών, όπως η RoleSimplifierCurator (βλ. RoleSimplifierCurator.cs), είναι ένα παράδειγμα συντηρησιμότητας και επεκτασιμότητας. Αυτές οι υπηρεσίες αναλαμβάνουν την τυποποίηση και τον καθαρισμό δεδομένων, καθιστώντας το σύστημα πιο ανθεκτικό σε ανομοιογενή εισαγωγή δεδομένων και επιτρέποντας την εύκολη προσθήκη νέων κανόνων καθαρισμού.
- **API Documentation (Swagger):** Η αυτόματη δημιουργία τεκμηρίωσης του API μέσω Swagger (βλ. Program.cs) διευκολύνει τους προγραμματιστές που αλληλεπιδρούν με το backend, βελτιώνοντας τη συντηρησιμότητα και την επεκτασιμότητα.

5. Υλοποίηση

Αφού ολοκληρώθηκε η θεωρητική θεμελίωση και ο αρχιτεκτονικός σχεδιασμός του συστήματος στα προηγούμενα κεφάλαια, το παρόν κεφάλαιο εστιάζει στη φάση της υλοποίησης. Εδώ, οι αφηρημένες σχεδιαστικές αρχές και οι απαιτήσεις μετατρέπονται σε απτό, λειτουργικό κώδικα. Η διαδικασία αυτή αποτελεί την καρδιά της πρακτικής ανάπτυξης, όπου οι τεχνολογικές επιλογές παίρνουν μορφή και το σύστημα αρχίζει να αποκτά τη λειτουργικότητά του. Στις ενότητες που ακολουθούν, θα παρουσιαστεί αναλυτικά η διαδικασία της υλοποίησης, ξεκινώντας από τα εργαλεία που χρησιμοποιήθηκαν, περιγράφοντας τη δομή του κώδικα τόσο στο backend όσο και στο frontend, και αναλύοντας συγκεκριμένα παραδείγματα υλοποίησης κρίσιμων λειτουργιών. Επιπλέον, θα καταγραφούν οι προκλήσεις που αναδείχθηκαν κατά την ανάπτυξη και οι λύσεις που εφαρμόστηκαν για την αντιμετώπισή τους.

5.1 Εργαλεία Ανάπτυξης

Η επιλογή των κατάλληλων εργαλείων ανάπτυξης αποτέλεσε κρίσιμο παράγοντα για την παραγωγικότητα, την ποιότητα του κώδικα και την αποτελεσματική διαχείριση του έργου. Για την υλοποίηση του συστήματος «Θεατρικό», χρησιμοποιήθηκε μια σουίτα σύγχρονων και καθιερωμένων εργαλείων που κάλυψαν ολόκληρο τον κύκλο ζωής της ανάπτυξης, από τη συγγραφή του κώδικα μέχρι τον έλεγχο και την αποσφαλμάτωση. Για την ανάπτυξη της υποστηρικτικής υποδομής (backend) σε ASP.NET Core, αξιοποιήθηκαν ολοκληρωμένα περιβάλλοντα ανάπτυξης (IDE) όπως το JetBrains Rider ή το Microsoft Visual Studio. Για τον έλεγχο και την τεκμηρίωση των endpoints του API, χρησιμοποιήθηκε εκτενώς το ενσωματωμένο περιβάλλον του Swagger (OpenAPI), καθώς και εξειδικευμένα εργαλεία όπως το Postman για την αποστολή δοκιμαστικών αιτημάτων HTTP. Στην πλευρά της εφαρμογής-πελάτη (frontend), η ανάπτυξη σε Flutter πραγματοποιήθηκε με τη χρήση του επεξεργαστή κώδικα Visual Studio Code, ο οποίος, σε συνδυασμό με τις επίσημες επεκτάσεις για Dart και Flutter, παρέχει ένα ισχυρό περιβάλλον με δυνατότητες όπως η ταχεία ανανέωση (Hot Reload). Για τη διαχείριση του πηγαίου κώδικα και τη συνεργατική ανάπτυξη, χρησιμοποιήθηκε το σύστημα ελέγχου εκδόσεων Git [3], [17].

5.2 Δομή Κώδικα

Η οργάνωση του πηγαίου κώδικα ακολούθησε καθιερωμένα πρότυπα και αρχές που διασφαλίζουν τη συντηρησιμότητα, την επεκτασιμότητα και την ευκρίνεια του έργου. Υιοθετήθηκε ένας σαφής διαχωρισμός μεταξύ της λογικής του backend και του frontend, με το καθένα να διαθέτει τη δική του διακριτή δομή.
Στο **backend**, το οποίο αναπτύχθηκε σε ASP.NET Core, η δομή του έργου ακολουθεί τις αρχές της Καθαρής Αρχιτεκτονικής (Clean Architecture). Οι βασικοί κατάλογοι περιλαμβάνουν τους Controllers, οι οποίοι αποτελούν το σημείο εισόδου για τα αιτήματα του API, τους Services, που περιέχουν την κύρια επιχειρησιακή λογική, και τους Repositories, που υλοποιούν το Repository Pattern για την αφαίρεση της λογικής πρόσβασης στη βάση δεδομένων. Μια ιδιαίτερα σημαντική ενότητα είναι ο

κατάλογος Curators, όπως φαίνεται στο αρχείο RoleSimplifierCurator.cs, ο οποίος περιέχει εξειδικευμένες κλάσεις υπεύθυνες για τον «καθαρισμό» και την τυποποίηση των εισερχόμενων δεδομένων, διασφαλίζοντας την ομοιογένειά τους. Η καταχώριση όλων αυτών των υπηρεσιών μέσω Dependency Injection στο αρχείο Program.cs επιτρέπει την εύκολη διαχείριση και αντικατάστασή τους [17], [12].
Στο **frontend**, η εφαρμογή Flutter είναι δομημένη γύρω από τον κεντρικό κατάλογο lib. Εντός αυτού, ο κατάλογος pages περιέχει τις οθόνες της εφαρμογής, οργανωμένες σε υποκαταλόγους ανάλογα με τη λειτουργικότητά τους (π.χ., home, theaters, user). Ο κατάλογος using φιλοξενεί βοηθητικές κλάσεις και υπηρεσίες, όπως το UserService.dart που διαχειρίζεται την επικοινωνία με το API για θέματα χρηστών, και το MyColors.dart που ορίζει την παλέτα χρωμάτων της εφαρμογής. Τέλος, ο κατάλογος models περιέχει τις κλάσεις-μοντέλα (π.χ., Theater.dart) που αναπαριστούν τις δομές δεδομένων που ανταλλάσσονται με το backend, επιτρέποντας την ισχυρή τυποποίηση (strong typing) και την αποφυγή σφαλμάτων [3].

5.3 Υλοποίηση API Calls

Η επικοινωνία μεταξύ της εφαρμογής Flutter (client) και του ASP.NET Core API (server) αποτελεί τον πυρήνα της λειτουργικότητας του συστήματος. Η υλοποίηση των κλήσεων API (API calls) στο frontend πραγματοποιήθηκε με τη χρήση του πακέτου http της Dart. Κάθε κλήση ακολουθεί μια τυποποιημένη διαδικασία: αρχικά, κατασκευάζεται το αντικείμενο Uri που αντιστοιχεί στο επιθυμητό endpoint του API. Στη συνέχεια, το αίτημα αποστέλλεται χρησιμοποιώντας την κατάλληλη μέθοδο HTTP (GET, POST, PUT, DELETE). Ένα κρίσιμο στοιχείο σε κάθε αίτημα είναι η προσθήκη του Authorization header, το οποίο περιέχει το JSON Web Token (JWT) του συνδεδεμένου χρήστη, διασφαλίζοντας την ταυτοποίηση και την εξουσιοδότησή του από το backend. Μετά την παραλαβή της απάντησης, ελέγχεται ο κωδικός κατάστασης (status code). Μια απάντηση με κωδικό 200 (OK) υποδηλώνει επιτυχία, οπότε το σώμα της απάντησης (response body), που είναι σε μορφή JSON, αποκωδικοποιείται και μετατρέπεται σε αντικείμενα Dart για περαιτέρω επεξεργασία και προβολή στο UI [16].
Ένα χαρακτηριστικό παράδειγμα είναι η συνάρτηση updateSocialMedia στην κλάση UserService.dart, η οποία αναλαμβάνει την ενημέρωση των κοινωνικών δικτύων του χρήστη.

```
static Future<bool> updateSocialMedia(String platform, String url) async {
  try {
    if (globalAccessToken == null) {
      print("❌ Δεν υπάρχει αποθηκευμένο token.");
      return false;
    }

    // ✅ Διορθωμένο: Το link περνάει ως Query Parameter και όχι στο body!
    Uri uri = Uri.parse(

"http://${Constants().hostName}/api/User/@/$platform?link=${Uri.encodeComponen
t(url)}");
```

```
    print("📤 Request προς API:");
    print("🔹 URL: $uri");

    http.Response response = await http.put(
      uri,
      headers: {
        "Authorization": "Bearer $globalAccessToken",
        "Content-Type": "application/json",
      },
    );

    if (response.statusCode == 200) {
      print("✅ Το $platform ενημερώθηκε επιτυχώς!");
      return true;
    } else {
      print("❌ Σφάλμα ενημέρωσης του $platform: ${response.statusCode}");
      print("📩 API Response: ${response.body}");
      return false;
    }
  } catch (e) {
    print("❌ Σφάλμα κατά την ενημέρωση του $platform: $e");
    return false;
  }
}
```

Στον παραπάνω κώδικα, η συνάρτηση κατασκευάζει το Uri για το endpoint /api/User/@/{platform}, προσθέτοντας το URL του κοινωνικού δικτύου ως query parameter. Στη συνέχεια, εκτελεί ένα αίτημα PUT, επισυνάπτοντας το JWT token στο header. Η επιτυχία ή αποτυχία της κλήσης καθορίζεται από τον statusCode της απάντησης του API.

5.4 Νέες Λειτουργίες

Πέρα από τις βασικές λειτουργίες ανάκτησης και παρουσίασης δεδομένων, κατά την υλοποίηση ενσωματώθηκαν καινοτόμες λειτουργίες που εμπλουτίζουν την εμπειρία χρήστη και ενισχύουν την ασφάλεια και την αξιοπιστία της πλατφόρμας. Μία από τις σημαντικότερες είναι το σύστημα **επαλήθευσης τηλεφώνου και αυθεντικοποίησης δύο παραγόντων (2FA)**. Η υλοποίηση αυτής της λειτουργίας συνδυάζει στοιχεία από το UI, την υπηρεσία χρήστη στο frontend και το backend. Στην οθόνη επεξεργασίας προφίλ (EditProfileScreen.dart), ο χρήστης έχει τη δυνατότητα να προσθέσει τον αριθμό τηλεφώνου του. Η εφαρμογή, μέσω της UserService, καλεί το API για να καταχωρίσει τον αριθμό. Στη συνέχεια, για την επαλήθευση, ο χρήστης μπορεί να ζητήσει την αποστολή ενός κωδικού μιας χρήσης (OTP). Η UserService καλεί το endpoint request-verification-phone-number, το οποίο με τη σειρά του, μέσω της ITwilioService που έχει δηλωθεί στο Program.cs, επικοινωνεί με την υπηρεσία Twilio για την αποστολή του SMS. Ο χρήστης εισάγει τον κωδικό στην εφαρμογή, ο οποίος αποστέλλεται για επιβεβαίωση στο

API. Μετά την επιτυχή επαλήθευση, ο χρήστης μπορεί να ενεργοποιήσει το 2FA, προσθέτοντας ένα επιπλέον επίπεδο ασφάλειας στον λογαριασμό του.

5.5 Δυσκολίες και Λύσεις

Κατά τη διάρκεια της ανάπτυξης, προέκυψαν ορισμένες τεχνικές προκλήσεις που απαιτούσαν την εξεύρεση αποδοτικών λύσεων. Μία από τις κυριότερες δυσκολίες ήταν η **διαχείριση της ασυνέπειας των δεδομένων** που προέρχονταν από εξωτερικές πηγές, ειδικά όσον αφορά τις ονομασίες των ρόλων των συντελεστών. Για παράδειγμα, ο ρόλος του βοηθού σκηνοθέτη εμφανιζόταν με δεκάδες διαφορετικές παραλλαγές, όπως «βοηθός σκηνοθέτη», «β. σκηνοθέτη» κ.λπ. Για την αντιμετώπιση αυτού του προβλήματος, υλοποιήθηκε στο backend μια εξειδικευμένη υπηρεσία, η RoleSimplifierCurator. Η λύση αυτή περιλαμβάνει τη δημιουργία ενός λεξικού (GetCorrectedRoleDictionary) που αντιστοιχίζει όλες τις γνωστές λανθασμένες ή εναλλακτικές γραφές σε μία, κανονικοποιημένη τιμή. Πριν την αποθήκευση οποιουδήποτε νέου ρόλου, η υπηρεσία αυτή παρεμβαίνει και «καθαρίζει» την τιμή, διασφαλίζοντας την ομοιογένεια και την ακεραιότητα των δεδομένων στη βάση.
Μια δεύτερη πρόκληση, χαρακτηριστική της ανάπτυξης σε Flutter, ήταν η **αποτελεσματική διαχείριση της κατάστασης (state management)** και η ανανέωση του UI μετά την ολοκλήρωση ασύγχρονων λειτουργιών, όπως οι κλήσεις στο API. Η λύση που υιοθετήθηκε βασίστηκε στη χρήση των StatefulWidget και της μεθόδου setState(). Όπως φαίνεται στο αρχείο TheaterInfo.dart, στη μέθοδο _loadData, τα δεδομένα για το θέατρο και το προφίλ του χρήστη ανακτώνται ασύγχρονα. Μόλις τα δεδομένα είναι διαθέσιμα, καλείται η setState(), η οποία δίνει εντολή στο Flutter framework να ξαναχτίσει το widget tree, προβάλλοντας έτσι τις νέες πληροφορίες στην οθόνη και αντικαθιστώντας την ένδειξη φόρτωσης [9].

5.6 Testing

Ο έλεγχος του συστήματος αποτέλεσε αναπόσπαστο και συνεχές μέρος της διαδικασίας υλοποίησης, με στόχο τη διασφάλιση της ορθής λειτουργίας, της σταθερότητας και της ποιότητας της εφαρμογής. Η στρατηγική ελέγχου περιελάμβανε πολλαπλά επίπεδα. Σε πρώτο επίπεδο, πραγματοποιήθηκε εκτενής **λειτουργικός έλεγχος του backend API** με τη χρήση εργαλείων όπως το Postman. Κάθε endpoint δοκιμάστηκε μεμονωμένα με διάφορα δεδομένα εισόδου για να επιβεβαιωθεί ότι επιστρέφει τους σωστούς κωδικούς κατάστασης, τα αναμενόμενα δεδομένα σε περιπτώσεις επιτυχίας, και τα κατάλληλα μηνύματα σφάλματος σε περιπτώσεις αποτυχίας. Παράλληλα, στο frontend, πραγματοποιήθηκε **χειροκίνητος έλεγχος (manual testing)**, όπου εκτελέστηκαν όλα τα σενάρια χρήσης που ορίστηκαν στο κεφάλαιο της ανάλυσης απαιτήσεων. Αυτό περιελάμβανε τον έλεγχο της πλοήγησης, της λειτουργικότητας των φορμών, της ορθής εμφάνισης των δεδομένων και της απόκρισης της εφαρμογής σε διαφορετικές συσκευές και αναλύσεις οθόνης. Τέλος, πραγματοποιήθηκε **έλεγχος ολοκλήρωσης (integration testing)**, όπου δοκιμάστηκε η ομαλή συνεργασία μεταξύ του frontend και του backend, διασφαλίζοντας ότι οι κλήσεις από την εφαρμογή Flutter λαμβάνονται και επεξεργάζονται σωστά από το API και ότι οι απαντήσεις του API ερμηνεύονται και προβάλλονται ορθά στο UI [19].

6. Αξιολόγηση & Testing

Η φάση της υλοποίησης, αν και κεντρική, δεν σηματοδοτεί το τέλος του κύκλου ανάπτυξης ενός πληροφοριακού συστήματος. Αντιθέτως, η αξιολόγηση και ο έλεγχος (testing) αποτελούν θεμελιώδη στάδια που διασφαλίζουν την ποιότητα, την αξιοπιστία και την ευθυγράμμιση του τελικού προϊόντος με τις αρχικές απαιτήσεις και τους στόχους. Το παρόν κεφάλαιο εστιάζει στη μεθοδολογία που ακολουθήθηκε για την επαλήθευση της λειτουργικότητας και της απόδοσης του «Θεατρικού Μητρώου Ελλάδας». Θα παρουσιαστούν αναλυτικά τα σενάρια και οι τύποι δοκιμών που εφαρμόστηκαν, τα αποτελέσματα που προέκυψαν, καθώς και η ανατροφοδότηση από τους χρήστες, η οποία είναι κρίσιμη για τη συνεχή βελτίωση της εφαρμογής.

6.1 Μεθοδολογία Testing

Η μεθοδολογία ελέγχου που εφαρμόστηκε για την εφαρμογή «Θεατρικό» υιοθέτησε μια συνδυαστική προσέγγιση, περιλαμβάνοντας τόσο χειροκίνητες όσο και αυτοματοποιημένες τεχνικές, ενσωματωμένες σε κάθε στάδιο του κύκλου ανάπτυξης. Ο στόχος ήταν η συστηματική αναγνώριση και διόρθωση σφαλμάτων, η επαλήθευση της ορθής λειτουργίας των απαιτήσεων και η διασφάλιση της συνολικής ποιότητας του συστήματος. Η διαδικασία ξεκίνησε από τα πρώτα στάδια της υλοποίησης, με ελέγχους σε επίπεδο μονάδας (unit testing) για τα επιμέρους συστατικά του backend, και κλιμακώθηκε σε ελέγχους ολοκλήρωσης (integration testing) και συστήματος (system testing), καταλήγοντας σε δοκιμές αποδοχής από τους χρήστες (User Acceptance Testing - UAT). Η επαναληπτική φύση της ανάπτυξης επέτρεψε τη συνεχή ανατροφοδότηση και την άμεση ενσωμάτωση διορθώσεων και βελτιώσεων [19].

6.2 Σενάρια Δοκιμών

Για την κάλυψη του εύρους λειτουργιών της εφαρμογής, σχεδιάστηκαν και εκτελέστηκαν συγκεκριμένα σενάρια δοκιμών, τα οποία βασίστηκαν στις λειτουργικές απαιτήσεις που καθορίστηκαν στο Κεφάλαιο 3. Τα σενάρια αυτά κάλυψαν τις βασικές ροές εργασίας των διαφόρων τύπων χρηστών (Θεατής, Επαγγελματίας, Διαχειριστής). Ενδεικτικά σενάρια περιελάμβαναν:

- **Εγγραφή και Σύνδεση Χρήστη:** Δοκιμάστηκε η επιτυχής δημιουργία νέου λογαριασμού, η σύνδεση με έγκυρα και μη έγκυρα διαπιστευτήρια, καθώς και η διαδικασία ανάκτησης κωδικού πρόσβασης.
- **Διαχείριση Προφίλ Χρήστη:** Ελέγχθηκε η δυνατότητα ενημέρωσης του ονόματος χρήστη, η προσθήκη και διαγραφή συνδέσμων κοινωνικών δικτύων, καθώς και η μεταφόρτωση και ρύθμιση φωτογραφίας προφίλ.
- **Επαλήθευση Τηλεφώνου και 2FA:** Δοκιμάστηκε η διαδικασία καταχώρισης αριθμού τηλεφώνου, η αποστολή και επιβεβαίωση του κωδικού OTP μέσω Twilio, καθώς και η ενεργοποίηση/απενεργοποίηση της επαλήθευσης δύο παραγόντων.
- **Πλοήγηση και Αναζήτηση Παραστάσεων:** Ελέγχθηκε η ορθή εμφάνιση των θεατρικών παραστάσεων, η λειτουργικότητα των φίλτρων (π.χ., ανά ημερομηνία, ανά είδος) και της αναζήτησης με λέξεις-κλειδιά.

- **Διεκδίκηση Προφίλ Συντελεστή:** Δοκιμάστηκε η υποβολή αιτήματος διεκδίκησης προφίλ ηθοποιού, η επισύναψη αποδεικτικών εγγράφων και η διαχείριση του αιτήματος από τον διαχειριστή (έγκριση/απόρριψη).
- **Διαχείριση Αγαπημένων:** Ελέγχθηκε η προσθήκη και αφαίρεση παραστάσεων από τη λίστα αγαπημένων του χρήστη.

6.3 Τύποι Testing

Για την ολοκληρωμένη αξιολόγηση του συστήματος, εφαρμόστηκαν οι ακόλουθοι τύποι δοκιμών [19]:

- **Unit Testing (Δοκιμές Μονάδας):** Πραγματοποιήθηκαν δοκιμές σε μεμονωμένες μονάδες κώδικα, κυρίως στο backend (ASP.NET Core), για την επαλήθευση της ορθής λειτουργίας συγκεκριμένων μεθόδων και κλάσεων, όπως οι υπηρεσίες επικύρωσης (IUserValidationService) και οι curators (IRoleSimplifierCurator). Αυτό διασφάλισε ότι κάθε συστατικό λειτουργεί σωστά μεμονωμένα.
- **Integration Testing (Δοκιμές Ολοκλήρωσης):** Ελέγχθηκε η ομαλή συνεργασία μεταξύ διαφορετικών μονάδων και συστατικών. Για παράδειγμα, δοκιμάστηκε η επικοινωνία μεταξύ της εφαρμογής Flutter και του ASP.NET Core API, η αλληλεπίδραση του API με τη βάση δεδομένων PostgreSQL μέσω Entity Framework Core, καθώς και η ενσωμάτωση με εξωτερικές υπηρεσίες όπως το Twilio για την αποστολή SMS.
- **System Testing (Δοκιμές Συστήματος):** Διεξήχθησαν δοκιμές στο πλήρως ολοκληρωμένο σύστημα για να επιβεβαιωθεί ότι όλες οι λειτουργικές και μη-λειτουργικές απαιτήσεις πληρούνται. Αυτό περιελάμβανε τον έλεγχο της συνολικής απόδοσης, της ασφάλειας και της σταθερότητας της εφαρμογής σε πραγματικές συνθήκες.
- **User Acceptance Testing (UAT - Δοκιμές Αποδοχής Χρήστη):** Ένας μικρός αριθμός πιθανών τελικών χρηστών συμμετείχε σε δοκιμές για να αξιολογήσει την εφαρμογή από τη δική του οπτική γωνία. Στόχος ήταν να επιβεβαιωθεί ότι το σύστημα ανταποκρίνεται στις πραγματικές ανάγκες και προσδοκίες τους, και ότι είναι εύχρηστο και διαισθητικό.
- **Performance Testing (Δοκιμές Απόδοσης):** Αν και όχι σε πλήρη κλίμακα, πραγματοποιήθηκαν βασικές δοκιμές απόδοσης για την αξιολόγηση των χρόνων απόκρισης του API και της ταχύτητας φόρτωσης της εφαρμογής. Ελέγχθηκε η συμπεριφορά του συστήματος υπό φόρτο, με έμφαση στην αποτελεσματικότητα των μηχανισμών caching.
- **Security Testing (Δοκιμές Ασφάλειας):** Ελέγχθηκαν οι μηχανισμοί αυθεντικοποίησης (JWT) και εξουσιοδότησης (Authorization Filters), η κρυπτογράφηση της επικοινωνίας (HTTPS) και η λειτουργικότητα του 2FA, για την αναγνώριση πιθανών ευπαθειών και τη διασφάλιση της προστασίας των δεδομένων.

6.4 Αποτελέσματα

Τα αποτελέσματα των δοκιμών υπήρξαν ενθαρρυντικά, επιβεβαιώνοντας την ορθή λειτουργία των βασικών χαρακτηριστικών του «Θεατρικού Μητρώου Ελλάδας» και την επίτευξη των περισσότερων απαιτήσεων.

Οι **λειτουργικές δοκιμές** έδειξαν ότι όλες οι προβλεπόμενες λειτουργίες, από την εγγραφή χρήστη και τη διαχείριση προφίλ έως την αναζήτηση παραστάσεων και το σύστημα διεκδίκησης, εκτελούνται με επιτυχία. Εντοπίστηκαν και διορθώθηκαν ορισμένα μικρά σφάλματα (bugs) που αφορούσαν κυρίως την οπτική παρουσίαση (UI glitches) και την οριακή συμπεριφορά σε συγκεκριμένα σενάρια.
Οι **δοκιμές απόδοσης** κατέδειξαν ότι το ASP.NET Core API ανταποκρίνεται με ικανοποιητικούς χρόνους, με τις περισσότερες κλήσεις να ολοκληρώνονται σε λιγότερο από 500ms, χάρη στην αποτελεσματική χρήση του Entity Framework Core και των μηχανισμών caching. Η εφαρμογή Flutter επέδειξε ομαλή πλοήγηση και γρήγορη φόρτωση περιεχομένου, αξιοποιώντας τις δυνατότητες του framework.
Οι **δοκιμές ασφάλειας** επιβεβαίωσαν την ορθή εφαρμογή των μηχανισμών JWT για την αυθεντικοποίηση και των Authorization Filters για τον έλεγχο πρόσβασης. Η λειτουργία 2FA, σε συνδυασμό με την επαλήθευση τηλεφώνου μέσω Twilio, προσέθεσε ένα σημαντικό επίπεδο ασφάλειας στους λογαριασμούς των χρηστών.
Συνολικά, το σύστημα αποδείχθηκε σταθερό και αξιόπιστο, ικανό να διαχειριστεί τις προβλεπόμενες λειτουργίες με αποτελεσματικότητα.

6.5 Feedback Χρηστών

Για την αξιολόγηση της ευχρηστίας και της συνολικής εμπειρίας χρήστη, πραγματοποιήθηκαν δοκιμές αποδοχής με μια μικρή ομάδα πιθανών τελικών χρηστών. Η ανατροφοδότηση που συλλέχθηκε ήταν πολύτιμη και παρείχε σημαντικές πληροφορίες για περαιτέρω βελτιώσεις [20].
Οι χρήστες εξέφρασαν γενικά την ικανοποίησή τους για την **αισθητική** και τη **διαισθητική πλοήγηση** της εφαρμογής. Ειδικότερα, εκτίμησαν την καθαρή διάταξη των πληροφοριών για τις παραστάσεις, την ευκολία στην αναζήτηση και το φιλτράρισμα, καθώς και την άμεση πρόσβαση σε λεπτομέρειες για τους συντελεστές και τους χώρους. Η λειτουργία των "Αγαπημένων" και η δυνατότητα κοινοποίησης σε κοινωνικά δίκτυα κρίθηκαν ως ιδιαίτερα χρήσιμες.
Ως προς τις **προτάσεις βελτίωσης**, αναφέρθηκαν ορισμένα σημεία:

- **Ενίσχυση της Προσωποποίησης:** Ζητήθηκε η δυνατότητα για πιο προηγμένες επιλογές εξατομίκευσης, όπως η λήψη ειδοποιήσεων για συγκεκριμένους συντελεστές ή θέατρα.
- **Βελτίωση της Διαχείρισης Εικόνων:** Προτάθηκε η προσθήκη λειτουργιών όπως η ομαδική μεταφόρτωση εικόνων ή η δημιουργία άλμπουμ στο προφίλ του χρήστη.
- **Ενημέρωση Περιεχομένου:** Επισημάνθηκε η ανάγκη για συνεχή και άμεση ενημέρωση του περιεχομένου, ειδικά όσον αφορά τις ημερομηνίες και τις τιμές των παραστάσεων.

Το feedback των χρηστών επιβεβαίωσε την ανάγκη για συνεχή εξέλιξη της εφαρμογής, με έμφαση στην περαιτέρω προσωποποίηση και την εμπλουτισμό των λειτουργιών διαχείρισης περιεχομένου.

6.6 Συγκρίσεις με Προηγούμενη Έκδοση

Δεδομένου ότι το «Θεατρικό» αποτελεί μια νέα, ολοκληρωμένη υλοποίηση, η σύγκριση με μια προηγούμενη έκδοση του ίδιου συστήματος δεν είναι άμεσα εφικτή. Ωστόσο, η αξιολόγηση μπορεί να γίνει σε σχέση με τους αρχικούς στόχους και τις απαιτήσεις που

τέθηκαν, καθώς και σε σύγκριση με το γενικότερο τοπίο των υφιστάμενων πολιτιστικών εφαρμογών στην Ελλάδα.
Σε σχέση με τους **αρχικούς στόχους**, το σύστημα επιτυγχάνει:

- **Κεντρικοποίηση Πληροφορίας:** Παρέχει ένα ενιαίο σημείο πρόσβασης σε κατακερματισμένες πληροφορίες για το θέατρο, κάτι που απουσίαζε σε αυτή την έκταση.
- **Δυναμική Διαχείριση Περιεχομένου:** Μέσω του συστήματος διεκδίκησης (claiming system), οι επαγγελματίες αποκτούν τον έλεγχο των προφίλ τους, κάτι που δεν προσφέρεται από τις περισσότερες απλές ενημερωτικές εφαρμογές.
- **Σύγχρονη Εμπειρία Χρήστη:** Η εφαρμογή Flutter προσφέρει μια σύγχρονη, αισθητικά ευχάριστη και διαισθητική εμπειρία, ξεπερνώντας σε πολλά σημεία την ευχρηστία παλαιότερων ή λιγότερο φροντισμένων εφαρμογών του χώρου.

Σε σύγκριση με **υφιστάμενες πολιτιστικές εφαρμογές** στην Ελλάδα, το «Θεατρικό» διακρίνεται για:

- **Ολοκληρωμένη Αρχιτεκτονική:** Η χρήση ενός στιβαρού backend (ASP.NET Core) και μιας σύγχρονης cross-platform τεχνολογίας (Flutter) το καθιστά τεχνολογικά πιο προηγμένο και επεκτάσιμο.
- **Εστίαση στην Ασφάλεια:** Η ενσωμάτωση 2FA και η χρήση JWT για την αυθεντικοποίηση το τοποθετούν σε υψηλότερο επίπεδο ασφάλειας σε σχέση με πολλές απλούστερες εφαρμογές.
- **Καινοτόμες Λειτουργίες:** Το σύστημα διεκδίκησης προφίλ αποτελεί μια καινοτομία που ενισχύει την αξιοπιστία των δεδομένων και την αλληλεπίδραση των επαγγελματιών με την πλατφόρμα.

Συμπερασματικά, η υλοποίηση όχι μόνο ανταποκρίθηκε στις αρχικές απαιτήσεις, αλλά έθεσε και τις βάσεις για ένα σύστημα που μπορεί να εξελιχθεί και να προσφέρει σημαντική αξία στον πολιτιστικό τομέα, υπερβαίνοντας τις δυνατότητες πολλών υφιστάμενων λύσεων.

7. Συμπεράσματα & Μελλοντική Εργασία

Η ολοκλήρωση ενός έργου ανάπτυξης λογισμικού δεν σηματοδοτεί μόνο την επίτευξη των αρχικών στόχων, αλλά και την ευκαιρία για μια αναδρομική αξιολόγηση της συνολικής προσπάθειας. Το παρόν κεφάλαιο αποτελεί το επιστέγασμα της διπλωματικής εργασίας, συνοψίζοντας τα βασικά επιτεύγματα και αναδεικνύοντας τη συμβολή του «Θεατρικού» στον τομέα των πολιτιστικών εφαρμογών. Παράλληλα, αναγνωρίζονται οι περιορισμοί που ενδεχομένως προέκυψαν κατά τη διάρκεια της υλοποίησης και, το σημαντικότερο, διατυπώνονται συγκεκριμένες προτάσεις για τη μελλοντική επέκταση και βελτίωση του συστήματος, ανοίγοντας νέους δρόμους για την περαιτέρω αξιοποίηση της τεχνολογίας στην προώθηση του θεάτρου.

7.1 Επιτεύγματα

Η παρούσα διπλωματική εργασία οδήγησε στην επιτυχή σχεδίαση και υλοποίηση του «Θεατρικού Μητρώου Ελλάδας», μιας ολοκληρωμένης ψηφιακής πλατφόρμας που γεφυρώνει τον κόσμο του ελληνικού θεάτρου με τις σύγχρονες τεχνολογίες κινητών εφαρμογών. Ένα από τα σημαντικότερα επιτεύγματα είναι η δημιουργία μιας cross-platform mobile εφαρμογής με χρήση του Flutter, η οποία παρέχει μια ενιαία και συνεκτική εμπειρία χρήστη τόσο σε συσκευές iOS όσο και Android, αποφεύγοντας την ανάγκη για ξεχωριστή ανάπτυξη και συντήρηση κώδικα. Παράλληλα, αναπτύχθηκε ένα στιβαρό και επεκτάσιμο RESTful API με ASP.NET Core, το οποίο διαχειρίζεται αποτελεσματικά την επιχειρησιακή λογική, την ασφάλεια και την επικοινωνία με τη βάση δεδομένων PostgreSQL και την υπηρεσία object storage Minio.
Επιπλέον, το σύστημα επιδεικνύει την επιτυχή ενσωμάτωση σύγχρονων μηχανισμών ασφαλείας, όπως η αυθεντικοποίηση μέσω JWT και η επαλήθευση δύο παραγόντων (2FA) με τη χρήση της υπηρεσίας Twilio, διασφαλίζοντας την προστασία των δεδομένων των χρηστών. Ένα ιδιαίτερο επίτευγμα αποτελεί η υλοποίηση του συστήματος διεκδίκησης (claiming system), το οποίο επιτρέπει σε επαγγελματίες του θεάτρου να αναλάβουν τον έλεγχο των ψηφιακών τους προφίλ ή των παραστάσεων στις οποίες συμμετέχουν, ενισχύοντας την αξιοπιστία και την επικαιροποίηση του περιεχομένου. Τέλος, η εφαρμογή προσφέρει μια διαισθητική και αισθητικά άρτια διεπαφή χρήστη, με λειτουργίες αναζήτησης, φιλτραρίσματος και εξατομίκευσης, καθιστώντας την πρόσβαση σε θεατρικές πληροφορίες πιο εύκολη και ελκυστική από ποτέ [14], [13].

7.2 Συμβολή της Εργασίας

Η συμβολή της παρούσας εργασίας είναι διττή, αγγίζοντας τόσο τον ακαδημαϊκό όσο και τον πρακτικό-πολιτιστικό τομέα. Από ακαδημαϊκής και τεχνολογικής σκοπιάς, η εργασία αποτελεί μια ολοκληρωμένη μελέτη περίπτωσης για την εφαρμογή σύγχρονων αρχιτεκτονικών προτύπων (Client-Server, Clean Architecture) και τεχνολογιών (Flutter, ASP.NET Core, PostgreSQL, Minio, Twilio) στην επίλυση ενός πραγματικού προβλήματος. Αναδεικνύει τις βέλτιστες πρακτικές στην ανάπτυξη mobile εφαρμογών, στην ασφάλεια των διαδικτυακών υπηρεσιών και στη διαχείριση ετερογενών δεδομένων, παρέχοντας ένα πρακτικό παράδειγμα για μελλοντικούς φοιτητές και ερευνητές [2].

Από πολιτιστικής και κοινωνικής σκοπιάς, το «Θεατρικό» συμβάλλει ενεργά στον ψηφιακό μετασχηματισμό του ελληνικού θεάτρου. Αντιμετωπίζει τον κατακερματισμό της πληροφορίας, προσφέροντας ένα κεντρικό σημείο αναφοράς για το θεατρόφιλο κοινό, τους επαγγελματίες του χώρου και τους ερευνητές. Μέσω της εύκολης πρόσβασης σε πληροφορίες για παραστάσεις, συντελεστές και θέατρα, η εφαρμογή προωθεί την πολιτιστική κληρονομιά και ενισχύει την προσβασιμότητα στην τέχνη, ιδίως για τις νεότερες γενιές που είναι εξοικειωμένες με τις ψηφιακές πλατφόρμες. Η δυνατότητα των καλλιτεχνών να διαχειρίζονται τα δικά τους προφίλ ενδυναμώνει την ψηφιακή τους παρουσία και ενισχύει την αλληλεπίδραση με το κοινό [1], [5].

7.3 Περιορισμοί

Παρά την επιτυχή υλοποίηση και τη σημαντική συμβολή της, η παρούσα εργασία παρουσιάζει ορισμένους περιορισμούς, οι οποίοι αποτελούν ταυτόχρονα σημεία εκκίνησης για μελλοντική έρευνα και ανάπτυξη. Ένας βασικός περιορισμός αφορά την έκταση των αυτοματοποιημένων δοκιμών. Αν και πραγματοποιήθηκαν εκτενείς χειροκίνητες δοκιμές και δοκιμές ολοκλήρωσης, η πλήρης κάλυψη με unit, widget και integration tests, ιδίως στο frontend, θα ενίσχυε περαιτέρω τη σταθερότητα και τη συντηρησιμότητα του κώδικα. Επιπλέον, η διαχείριση της κατάστασης (state management) στο Flutter, αν και λειτουργική, θα μπορούσε να βελτιστοποιηθεί με την υιοθέτηση πιο προηγμένων προτύπων, όπως το BLoC ή το Riverpod, για την αντιμετώπιση της πολυπλοκότητας σε μεγαλύτερες εφαρμογές.
Ένας άλλος περιορισμός σχετίζεται με την πληρότητα του περιεχομένου. Παρόλο που η βάση δεδομένων εμπλουτίστηκε με ένα αντιπροσωπευτικό σύνολο δεδομένων, η πλήρης κάλυψη του ελληνικού θεατρικού τοπίου απαιτεί συνεχή εισαγωγή και επικαιροποίηση πληροφοριών, μια διαδικασία που ξεπερνά τα όρια μιας διπλωματικής εργασίας. Τέλος, η απουσία ενός web-based διαχειριστικού περιβάλλοντος για τους διαχειριστές και τους Claims Managers καθιστά τη διαχείριση του συστήματος λιγότερο φιλική και πιο τεχνική, περιορίζοντας την ευκολία χρήσης για μη-προγραμματιστές.

7.4 Προτάσεις για Μελλοντική Εργασία

Βασιζόμενοι στους αναγνωρισμένους περιορισμούς και τις δυνατότητες επέκτασης του συστήματος, διατυπώνονται οι ακόλουθες προτάσεις για μελλοντική εργασία:

- **Επέκταση των Αυτοματοποιημένων Δοκιμών:** Η ανάπτυξη ενός ολοκληρωμένου πλαισίου αυτοματοποιημένων δοκιμών (unit, widget, integration tests) για το Flutter frontend και την επέκταση των unit tests στο ASP.NET Core backend θα βελτιώσει σημαντικά την ποιότητα του κώδικα και θα διευκολύνει τη μελλοντική ανάπτυξη [19].
- **Βελτιστοποίηση State Management στο Flutter:** Η αναβάθμιση του μηχανισμού διαχείρισης κατάστασης στο Flutter με τη χρήση πιο εξελιγμένων λύσεων, όπως το BLoC (Business Logic Component) ή το Riverpod, θα προσφέρει καλύτερη διαχωρισμό αρμοδιοτήτων, ευκολότερη δοκιμασιμότητα και βελτιωμένη απόδοση για σύνθετα σενάρια.
- **Ανάπτυξη Web Admin Panel:** Η δημιουργία ενός φιλικού προς τον χρήστη web-based διαχειριστικού περιβάλλοντος θα επιτρέψει στους διαχειριστές και τους

Claims Managers να διαχειρίζονται το περιεχόμενο, τους χρήστες και τα αιτήματα διεκδίκησης με μεγαλύτερη ευκολία και αποτελεσματικότητα.

- **Εμπλουτισμός Κοινωνικών Λειτουργιών:** Η προσθήκη λειτουργιών όπως σχόλια και αξιολογήσεις για παραστάσεις, δυνατότητα δημιουργίας προσωπικών λιστών παρακολούθησης (watchlists) και η ενσωμάτωση πιο προηγμένων μηχανισμών προσωποποίησης (π.χ., προτάσεις παραστάσεων βάσει ιστορικού και προτιμήσεων) θα ενισχύσει την αλληλεπίδραση των χρηστών.
- **Ενσωμάτωση Λειτουργιών Εισιτηρίων:** Η πλήρης ενσωμάτωση με συστήματα έκδοσης ηλεκτρονικών εισιτηρίων, επιτρέποντας στους χρήστες να αγοράζουν εισιτήρια απευθείας μέσω της εφαρμογής, θα προσέθετε σημαντική αξία.
- **Επέκταση Πολυμεσικού Περιεχομένου:** Η ενσωμάτωση περισσότερων πολυμεσικών στοιχείων, όπως trailers παραστάσεων, συνεντεύξεις συντελεστών και ψηφιακά αρχεία προγραμμάτων, θα εμπλουτίσει την εμπειρία του χρήστη [7].

7.5 Συμπεράσματα

Συνοψίζοντας, η παρούσα διπλωματική εργασία πέτυχε τον αρχικό της σκοπό, δημιουργώντας ένα λειτουργικό και τεχνολογικά σύγχρονο «Θεατρικό». Η υλοποίηση αυτή όχι μόνο επιβεβαίωσε τη σκοπιμότητα της αξιοποίησης των mobile τεχνολογιών στον πολιτιστικό τομέα, αλλά και ανέδειξε τις δυνατότητες για περαιτέρω καινοτομία. Το έργο αποτελεί μια ισχυρή βάση για την ψηφιοποίηση και την προώθηση του ελληνικού θεάτρου, προσφέροντας ένα εργαλείο που μπορεί να εξελιχθεί σε έναν κεντρικό κόμβο πληροφορίας και αλληλεπίδρασης για ολόκληρο το θεατρικό οικοσύστημα. Με τις προτεινόμενες μελλοντικές επεκτάσεις, το «Θεατρικό» μπορεί να αναδειχθεί σε ένα πρότυπο για την ενσωμάτωση της τεχνολογίας στην υπηρεσία του πολιτισμού [2], [20].

ΒΙΒΛΙΟΓΡΑΦΙΑ

Οι βιβλιογραφικές αναφορές παρατίθενται με τη σειρά που εμφανίζονται στο κείμενο, σύμφωνα με το πρότυπο IEEE.

[1] B. Hanussek, "Enhanced exhibitions? Discussing museum apps after a decade of development," Advances in Archaeological Practice, vol. 8, no. 2, pp. 206-212, 2020. doi: 10.1017/aap.2020.10.

[2] A. Biorn-Hansen, C. Rieger, T.-M. Gronli, T. A. Majchrzak and G. Ghinea, "An empirical investigation of performance overhead in cross-platform mobile development frameworks," Empirical Software Engineering, vol. 25, no. 4, pp. 2997-3040, 2020. doi: 10.1007/s10664-020-09827-6.

[3] Google LLC, "Flutter architectural overview," Flutter Documentation. [Online]. Available: https://docs.flutter.dev/resources/architectural-overview. [Accessed: Sep. 4, 2026].

[4] International Organization for Standardization, ISO 9241-11:2018, Ergonomics of human-system interaction - Part 11: Usability: Definitions and concepts. Geneva, Switzerland: ISO, 2018.

[5] U. Stankovic Elesini et al., "Mobile serious game for enhancing user experience in museum," ACM Journal on Computing and Cultural Heritage, vol. 16, no. 1, pp. 1-26, 2022. doi: 10.1145/3569088.

[6] A. Biorn-Hansen, T.-M. Gronli, G. Ghinea and S. Alouneh, "An empirical study of cross-platform mobile development in industry," Wireless Communications and Mobile Computing, vol. 2019, art. no. 5743892, pp. 1-12, 2019. doi: 10.1155/2019/5743892.

[7] C. Wang and Y. Zhu, "A survey of museum applied research based on mobile augmented reality," Computational Intelligence and Neuroscience, vol. 2022, art. no. 2926241, 2022. doi: 10.1155/2022/2926241.

[8] Apple Inc., "Human Interface Guidelines," Apple Developer Documentation. [Online]. Available: https://developer.apple.com/design/human-interface-guidelines/. [Accessed: Sep. 4, 2026].

[9] Google LLC, "Dart language documentation." [Online]. Available: https://dart.dev/language. [Accessed: Sep. 4, 2026].

[10] J. Nielsen, "Enhancing the explanatory power of usability heuristics," in Proc. SIGCHI Conf. Human Factors in Computing Systems (CHI 1994), Boston, MA, USA, 1994, pp. 152-158. doi: 10.1145/191666.191729.

[11] Google LLC, "Material Design 3 guidelines." [Online]. Available: https://m3.material.io/. [Accessed: Sep. 4, 2026].

[12] M. Fowler, Patterns of Enterprise Application Architecture. Boston, MA, USA: Addison-Wesley, 2002.

[13] OWASP Foundation, "OWASP Mobile Application Security Verification Standard (MASVS)." [Online]. Available: https://mas.owasp.org/MASVS/. [Accessed: Sep. 4, 2026].

[14] M. Jones, J. Bradley and N. Sakimura, "JSON Web Token (JWT)," RFC 7519, Internet Engineering Task Force, May 2015. doi: 10.17487/RFC7519.

[15] PostgreSQL Global Development Group, "PostgreSQL 15 documentation." [Online]. Available: https://www.postgresql.org/docs/15/. [Accessed: Sep. 4, 2026].

[16] R. T. Fielding, "Architectural styles and the design of network-based software architectures," Ph.D. dissertation, Dept. Inf. Comput. Sci., Univ. of California, Irvine, CA, USA, 2000.

[17] Microsoft Corporation, "ASP.NET Core documentation," Microsoft Learn. [Online]. Available: https://learn.microsoft.com/aspnet/core/. [Accessed: Sep. 4, 2026].

[18] Microsoft Corporation, "Entity Framework Core documentation," Microsoft Learn. [Online]. Available: https://learn.microsoft.com/ef/core/. [Accessed: Sep. 4, 2026].

[19] G. J. Myers, C. Sandler and T. Badgett, The Art of Software Testing, 3rd ed. Hoboken, NJ, USA: Wiley, 2011.

[20] J. Brooke, "SUS: A quick and dirty usability scale," in Usability Evaluation in Industry, P. W. Jordan, B. Thomas, B. A. Weerdmeester and I. L. McClelland, Eds. London, U.K.: Taylor and Francis, 1996, pp. 189-194.