ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«Ανάπτυξη διαδραστικής εφαρμογής για μικρό τεχνολογικό μουσείο»

Του φοιτητή
Χατζηγρηγορίου Σωτηρίου
Αρ. Μητρώου: 2019/187

Επιβλέπων
Ιορδάνης Κιοσκερίδης
Καθηγητής

Μάιος 2025

Ανάπτυξη διαδραστικής εφαρμογής για μικρό τεχνολογικό μουσείο
Κωδικός Δ.Ε.: 25127
Ονοματεπώνυμο φοιτητή: Σωτήριος Χατζηγρηγορίου
Ονοματεπώνυμο εισηγητή: Ιορδάνης Κιοσκερίδης
Ημερομηνία ανάληψης Δ.Ε. : 26/02/2025
Ημερομηνία περάτωσης Δ.Ε. : 28/05/2025

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

Περίληψη

Η παρούσα εργασία αφορά τη δημιουργία μιας εφαρμογής, η οποία δημιουργήθηκε για το μικρό τεχνολογικό μουσείο του Διεθνούς Πανεπιστημίου της Ελλάδας, που βρίσκεται στην πανεπιστημιούπολη της Σίνδου. Η ιδέα ήταν να ενισχυθεί η εμπειρία του επισκέπτη, χωρίς να αντικαθίσταται η φυσική επίσκεψη. Δηλαδή, να του παρέχεται η πληροφορία όταν τη χρειάζεται, χωρίς περιττά βήματα ή δύσκολες λειτουργίες.

Η βασική λειτουργία της εφαρμογής είναι απλή και λειτουργική. Δίπλα από κάθε έκθεμα υπάρχει ένας QR κωδικός. Ο επισκέπτης τον σαρώνει με το κινητό του, μέσω της εφαρμογής και βλέπει στην οθόνη μια περιγραφή για το αντίστοιχο έκθεμα. Το κείμενο είναι γραμμένο με κατανοητό τρόπο, χωρίς περίπλοκη ορολογία. Δεν έχει σημασία αν ο επισκέπτης είναι ειδικός ή απλά περαστικός. Ο στόχος είναι να καταλάβει σχετικά γρήγορα τι βλέπει.

Μαζί με την περιγραφή υπάρχουν και ερωτήσεις πολλαπλής επιλογής. Όχι για να βαθμολογηθεί κανείς, αλλά για να θυμηθεί, να συμμετάσχει, να δοκιμάσει τη γνώση του. Υπάρχει και κουίζ με ερωτήσεις από όλο το μουσείο, πιο γενικό και πιο χαλαρό, σαν παιχνίδι, όχι σαν διαγώνισμα.

Η εφαρμογή έχει δύο γλώσσες: ελληνικά και αγγλικά. Έγινε προσπάθεια να μεταφραστεί το περιεχόμενο των ερωτήσεων και οι απαντήσεις τους αυτόματα με τη βοήθεια του MyMemory API. Δεν είναι τέλεια η απόδοση, αλλά είναι αρκετή ώστε ένας ξένος επισκέπτης να καταλάβει βασικά πράγματα.

Επιπλέον, υπάρχει αναζήτηση εκθεμάτων με βάση το όνομά τους. Αν ο επισκέπτης θέλει να ξαναδεί κάτι ή να το βρει πιο εύκολα, μπορεί να το κάνει χωρίς να περιηγείται σε όλη την εφαρμογή. Τα εκθέματα είναι επίσης κατηγοριοποιημένα, για να υπάρχει τάξη και να ξέρει ο χρήστης τι είδους εκθέματα θέλει να αναζητήσει.

Μέσα στην εφαρμογή έχουν προστεθεί και απλές οδηγίες, για όσους δεν είναι εξοικειωμένοι με τέτοιου είδους εφαρμογές. Ο στόχος ήταν να μπορεί να τη χρησιμοποιήσει οποιοσδήποτε, ανεξαρτήτως ηλικίας ή τεχνολογικής εμπειρίας.

Από τεχνικής πλευράς, χρησιμοποιήθηκε το Flutter. Αυτό δίνει τη δυνατότητα να δημιουργηθεί η εφαρμογή με έναν κώδικα και να λειτουργεί σε Android αλλά και σε web περιβάλλον. Για την αποθήκευση των δεδομένων επιλέχθηκε το Supabase, το οποίο στηρίζεται στην PostgreSQL. Είναι σταθερό και καλύπτει τις ανάγκες της εφαρμογής χωρίς να προσθέτει επιπλοκές.

Γενικά, σκοπός δεν ήταν να εντυπωσιάσει η εφαρμογή και να καλύψει το περιεχόμενο του μουσείου και κατ’ επέκταση το ίδιο το μουσείο. Ήταν να είναι χρήσιμη. Να βοηθήσει τον επισκέπτη να καταλάβει καλύτερα τι βλέπει και να φύγει έχοντας κερδίσει κάτι. Αν το πετυχαίνει αυτό, τότε ο στόχος της έχει καλυφθεί.

«Development of an interactive application for a small technological museum»

Sotirios Chatzigrigoriou

Abstract

This project focuses on the creation of an application, developed specifically for the small technological museum of the International Hellenic University, located on the university campus in Sindos. The idea behind it was to enhance the visitor's experience without replacing the physical visit. In other words, to provide information at the moment it's needed, without unnecessary steps or complicated functions.

The core functionality of the app is simple and practical. Next to each exhibit, there is a QR code. The visitor scans it using their mobile device through the application, and immediately sees a description of the corresponding exhibit on their screen. The text is written in a clear and accessible way, avoiding complex terminology. It doesn't matter whether the visitor is an expert or just someone casually browsing — the goal is for them to quickly understand what they're looking at.

Along with the description, there are also multiple-choice questions. Not for scoring or grading, but to help the visitor remember, engage, and test what they know. There's also a general quiz with questions from the entire museum, more relaxed in nature — designed more like a game than a test.

The application supports two languages: Greek and English. An effort was made to automatically translate the content of the questions and answers using the MyMemory API. While the result may not be perfect, it's sufficient for a foreign visitor to understand the essential information.

In addition, the user can search for exhibits by name. If a visitor wants to revisit something or find it more easily, they can do so without having to browse the entire app. Exhibits are also categorized, to provide structure and help the user know what kind of items they're looking for.

Simple instructions have been added inside the app as well, for users who may not be familiar with such applications. The aim was to make it usable by anyone, regardless of age or technological experience.

From a technical point of view, the app was developed using Flutter. This allows it to be created with a single codebase and to function on both Android and web platforms. For data storage, Supabase was chosen, which is built on PostgreSQL. It's stable and covers the needs of the application without adding complexity.

Overall, the goal of the app was not to impress or to replace the content — or the presence — of the museum. It was designed to be useful. To help the visitor understand what they're seeing and leave having gained something more. If the app manages to do that, then its purpose has been fulfilled.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τους γονείς και την κοπέλα μου για την στήριξη και την υπομονή τους καθώς και τον καθηγητή μου για την καθοδήγηση του για την ολοκλήρωση της διπλωματικής εργασίας μου.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Συντομογραφίες

QR	Quick Response (μορφή δισδιάστατου γραμμωτού κώδικα)
API	Application Programming Interface (Διεπαφή Προγραμματισμού Εφαρμογών)
UI	User Interface (Διεπαφή Χρήστη)
UX	User Experience (Εμπειρία Χρήστη)
ORM	Object-Relational Mapping (Αντικειμενοσχεσιακή χαρτογράφηση)
PDF	Portable Document Format (Φορητή Μορφή Εγγράφου)
MVP	Minimum Viable Product (Ελάχιστο Βιώσιμο Προϊόν)
AR	Augmented Reality (Επαυξημένη Πραγματικότητα)
DB	Database (Βάση Δεδομένων)
BaaS	Backend as a Service (Υποδομή Πίσω Μέρους ως Υπηρεσία)
SDK	Software Development Kit (Σύνολο εργαλείων ανάπτυξης λογισμικού)

Κεφάλαιο 1ο: Εισαγωγή

1.1 Εισαγωγή

Τα τελευταία χρόνια, φαίνεται καθαρά ότι πολλά μουσεία αρχίζουν να χρησιμοποιούν εφαρμογές για κινητά και μάλλον όχι άδικα. Ο σημερινός επισκέπτης δεν είναι πια ικανοποιημένος μόνο με μια ταμπέλα κάτω από ένα έκθεμα. Θέλει να μπει στη διαδικασία να μάθει, να αλληλεπιδράσει, να ζήσει κάτι πιο ενεργό. Ιδίως στα τεχνολογικά μουσεία, που έχουν περιεχόμενο πιο δύσκολο να εξηγηθεί απλά, το κινητό μπορεί να γίνει βασικό εργαλείο. Και επειδή πολλά μικρά μουσεία δεν έχουν προσωπικό ή την οικονομική δυνατότητα για ξεναγήσεις ή πολύπλοκες τεχνολογικές λύσεις, μια εφαρμογή στο κινητό είναι η ιδανική λύση.

Το θετικό είναι ότι κάτι τέτοιο δεν απαιτεί ούτε ακριβό εξοπλισμό, ούτε τεχνικές γνώσεις. Το μόνο που χρειάζεται είναι ένα κινητό, το οποίο, σχεδόν όλοι έχουν. Ο επισκέπτης μπορεί να κινείται μόνος του, με τον δικό του ρυθμό, να σαρώνει, να διαβάζει και να απαντά σε ερωτήσεις. Δεν περιμένει οδηγίες, δεν ακολουθεί αυστηρό πρόγραμμα. Μέσα από αυτή τη σκέψη σχεδιάστηκε και η εφαρμογή που παρουσιάζεται σε αυτή την εργασία, ειδικά για το μικρό τεχνολογικό μουσείο του ΔΙΠΑΕ, που βρίσκεται στην πανεπιστημιούπολη της Σίνδου.

Για την υλοποίησή της χρησιμοποιήθηκε το Supabase, ώστε να αποθηκεύονται τα εκθέματα και οι ερωτήσεις, και το Flutter, για να μπορεί η εφαρμογή να λειτουργεί με τον ίδιο τρόπο και σε Android και μέσω browser, χωρίς ανάπτυξη ξεχωριστού κώδικα για κάθε μια από τις δύο πλατφόρμες καθώς και για σχετικά εύκολη επέκταση και σε ακόμα περισσότερες πλατφόρμες. Δηλαδή, με μία βάση κώδικα μπορεί να δημιουργηθεί μια εφαρμογή που θα τρέχει παντού, και αυτό είναι σημαντικό για λόγους χρόνου και συντήρησης.

Μέσα από την εφαρμογή, ο επισκέπτης μπορεί:

- Να σαρώνει τον QR κωδικό δίπλα σε ένα έκθεμα και να δει πληροφορίες στην οθόνη του.
- Να απαντήσει σε ερωτήσεις σχετικές με το αντικείμενο σαν παιχνίδι γνώσης.
- Να αλλάξει γλώσσα ανάλογα με το τι τον βολεύει — ελληνικά ή αγγλικά.
- Να δει τα εκθέματα ανά θεματική κατηγορία, αν θέλει να επικεντρωθεί σε κάτι συγκεκριμένο.

Στην κατασκευή της εφαρμογής, βασικός στόχος ήταν:

- Να μην είναι περίπλοκη. Ό,τι χρειάζεται ο επισκέπτης, να το βρίσκει εύκολα.
- Να υπάρχει τρόπος συμμετοχής, όχι μόνο παρατήρησης.
- Να μπορούν να τη χρησιμοποιήσουν και όσοι δεν ξέρουν ελληνικά.
- Να ενημερώνεται από απόσταση, χωρίς να αλλάζει μεγάλο μέρος του κώδικα κάθε φορά.

Κάτι που ξεχωρίζει είναι ότι το μουσείο μπορεί να προσθέσει νέο περιεχόμενο εύκολα. Αν εμφανιστεί ένα καινούριο έκθεμα, αρκεί να προστεθεί στο σύστημα. Δεν χρειάζεται να αλλάξει κάτι ιδιαίτερο στον κώδικα της εφαρμογής. Μπαίνει η περιγραφή, προστίθενται οι ερωτήσεις, και την επόμενη φορά που θα τη χρησιμοποιήσει κάποιος, όλα είναι ήδη εκεί.

Αυτή η προσέγγιση δεν έρχεται να ακυρώσει τα παραδοσιακά μουσεία. Αντίθετα, προσπαθεί να τα φέρει πιο κοντά σε αυτό που περιμένει ο σημερινός επισκέπτης. Ειδικά ο νέος επισκέπτης, που έχει συνηθίσει να χρησιμοποιεί το κινητό του όλο και πιο συχνά. Το να κοιτά ένα αντικείμενο και να διαβάζει απλώς μερικές πληροφορίες, δεν του φτάνει.

Η εφαρμογή που δημιουργήθηκε προσφέρει:

- **Σάρωση QR** και εμφάνιση βασικών πληροφοριών για κάθε έκθεμα.
- **Ερωτήσεις-κουίζ** που λειτουργούν σαν τρόπος συμμετοχής.
- **Αυτόματη μετάφραση των ερωτήσεων** μέσω του MyMemory API, για να μπορεί να χρησιμοποιείται και από μη ελληνόφωνους επισκέπτες.
- **Οργάνωση σε κατηγορίες** και αναζήτηση για ευκολότερη πλοήγηση.

Ο στόχος της εφαρμογής δεν ήταν να δείξει κάτι εντυπωσιακό. Ο στόχος ήταν να είναι χρήσιμη. Αν ένας επισκέπτης φύγει και έχει μάθει κάτι παραπάνω απ' ό,τι θα μάθαινε χωρίς αυτήν, τότε έχει πετύχει αυτό που σχεδιάστηκε να κάνει.

1.2 Δομή της εργασίας

Η εργασία ακολουθεί μια σαφή και δομημένη προσέγγιση για την ανάπτυξη της εφαρμογής και την ανάλυση των τεχνολογιών που χρησιμοποιήθηκαν. Η διάρθρωση της εργασίας περιλαμβάνει, εκτός από την περίληψη στα ελληνικά και στα αγγλικά όπου παρουσιάζονται συνοπτικά οι στόχοι και τα αποτελέσματα της εργασίας, πέντε βασικά κεφάλαια. Το πρώτο κεφάλαιο περιλαμβάνει την εισαγωγή, δηλαδή την ανάλυση του προβλήματος και την παρουσίαση του σκοπού της εφαρμογής.

Στο δεύτερο κεφάλαιο παρουσιάζεται το τεχνολογικό υπόβαθρο και παρόμοιες εφαρμογές που συνδέονται με το θέμα. Συγκεκριμένα, γίνεται αναφορά στη χρήση της τεχνολογίας QR, με την οποία γίνεται η αναγνώριση των εκθεμάτων μέσω κάμερας και πώς αυτή βελτιώνει την εμπειρία των επισκεπτών. Επίσης, περιγράφονται παρόμοιες εφαρμογές που χρησιμοποιούνται σε μουσεία όπως, για παράδειγμα, το Rijksmuseum στο Άμστερνταμ, το Βρετανικό μουσείο του Λονδίνου κ.α.

Στο τρίτο κεφάλαιο ακολουθεί η παρουσίαση του περιβάλλοντος ανάπτυξης και τα εργαλεία που χρησιμοποιήθηκαν για την υλοποίηση της εφαρμογής. Καταγράφονται οι λειτουργικές απαιτήσεις (τι πρέπει να κάνει η εφαρμογή) και το πώς αυτά υλοποιήθηκαν, συμπεριλαμβανομένης της σχεδίασης της βάσης δεδομένων και των κύριων διεπαφών χρήστη. Επιπλέον, περιγράφονται οι επιλογές σχεδιασμού που υιοθετήθηκαν, όπως η απόφαση για δυναμική μετάφραση του περιεχομένου και η δομή πλοήγησης της εφαρμογής.

Στο τέταρτο κεφάλαιο περιγράφεται αναλυτικά η υλοποίηση της εφαρμογής. Παρουσιάζεται η δομή του κώδικα της εφαρμογής και η υλοποίηση της βάσης δεδομένων στο Supabase. Επίσης, αναλύονται οι βασικές λειτουργίες μέσα από παραδείγματα κώδικα και εικόνες , παραδείγματα χρήσης της εφαρμογής , όπως η διαδικασία σάρωσης του QR κώδικα και ανάκτησης πληροφοριών, η διαδικασία μετάφρασης περιεχομένου σε δεύτερη γλώσσα και η λογική του συστήματος κουίζ. Συνοδευτικά σχήματα και πίνακες χρησιμοποιούνται για την καλύτερη κατανόηση των τεχνικών λεπτομερειών.

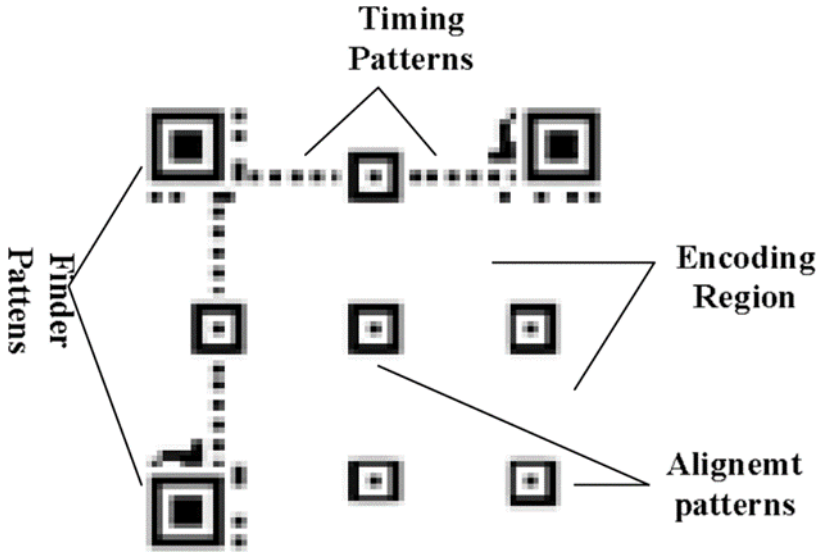
Τέλος, στο πέμπτο κεφάλαιο παρουσιάζονται τα συμπεράσματα που προκύπτουν από την ανάπτυξη και τη λειτουργία της εφαρμογής, καθώς και προτάσεις για μελλοντική εξέλιξη. Συζητείται ο βαθμός στον οποίο επιτεύχθηκαν οι στόχοι που τέθηκαν, τα οφέλη που αποκόμισε το μουσείο από την υλοποίηση της εφαρμογής, καθώς και πιθανές βελτιώσεις ή επεκτάσεις που θα μπορούσαν να υλοποιηθούν σε μελλοντικό στάδιο για να καταστεί η εφαρμογή ακόμη πιο αποτελεσματική και πλούσια σε λειτουργικότητα.

Η εργασία τεκμηριώνεται με εικόνες από την εφαρμογή και παραδείγματα κώδικα που αναδεικνύουν τα τεχνικά χαρακτηριστικά της ανάπτυξης.

Κεφάλαιο 2ο: Τεχνολογικό υπόβαθρο και παρόμοιες εφαρμογές

2.1 Εισαγωγή στην τεχνολογία QR

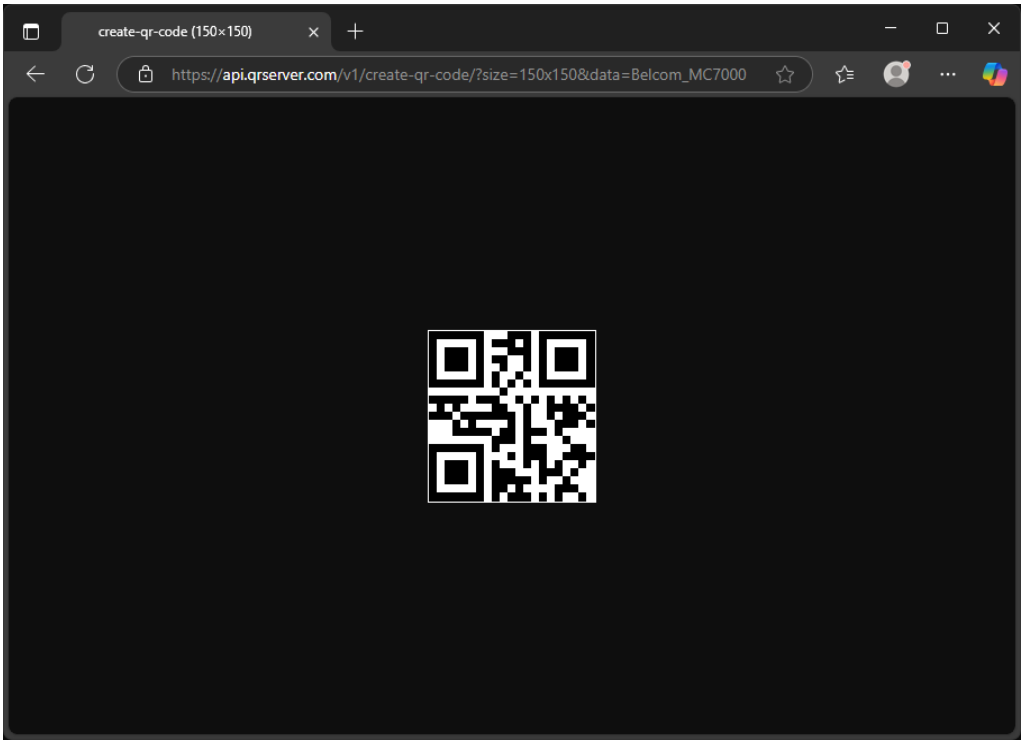
Τα τελευταία χρόνια συναντάται όλο και περισσότερο η χρήση QR κωδικών σε πολλούς τομείς της καθημερινότητας. Χρησιμοποιούνται σε ταυτοποίηση εισιτηρίων, εγγράφων, ταυτοτήτων καθώς και σε πραγματοποίηση πληρωμών. Το κύριο πλεονέκτημα τους είναι η μεγάλη ταχύτητα αναγνώρισης και αποκωδικοποίησης τους από την κάμερα ενός κοινού smartphone. Άλλωστε από αυτή την ιδιότητα, βγαίνει και το όνομα τους QR που είναι η συντομογραφία του Quick Response (γρήγορη απόκριση στα ελληνικά) [1][2]. Οι κωδικοί QR δημιουργήθηκαν από την εταιρεία Denso-Wave το 1994 και χρησιμοποιήθηκαν για τον έλεγχο των αποθεμάτων από ανταλλακτικά οχημάτων. Ακόμα ένα πλεονέκτημα των κωδικών QR είναι η μεγάλη χωρητικότητα τους. Μπορούν να περιέχουν 7089 χαρακτήρες σε αριθμούς και 4296 χαρακτήρες για αλφαριθμητικά [2]. Αυτό επιτυγχάνεται με την κωδικοποίηση των δεδομένων και στις δύο πλευρές (οριζόντια και κάθετη) [3]. Αυτό σημαίνει πως η συσκευή, με την οποία προσπαθούμε να σαρώσουμε έναν κωδικό QR, θα τον αναγνωρίσει σωστά, από όποια πλευρά και αν σαρωθεί. Η αρχιτεκτονική των κωδικών QR περιλαμβάνει τη λειτουργία αναζήτησης, τη λειτουργία χρονισμού και τη λειτουργία ευθυγράμμισης [4]. Υπάρχουν τρεις κοινές δομές στις τρεις γωνίες του συμβόλου ενός κώδικα QR που ονομάζονται μοτίβα αναζήτησης. Η λειτουργία εύρεσης χρησιμοποιείται για τον προσδιορισμό του σωστού προσανατολισμού ενός συμβόλου. Το λογισμικό αποκωδικοποίησης χρησιμοποιεί τη λειτουργία χρονισμού για να προσδιοριστεί η σωστή πλευρά του σχήματος. Εάν η λήψη από την κάμερα φαίνεται παραμορφωμένη, τότε χρησιμοποιείται το μοτίβο ευθυγράμμισης για να είναι σίγουρο ότι το λογισμικό αποκωδικοποίησης θα αποκωδικοποιήσει σωστά τον κωδικό. Η υπόλοιπη περιοχή εκτός από τον τρόπο λειτουργίας είναι μια περιοχή κωδικοποίησης που αποθηκεύει κωδικές λέξεις δεδομένων και κωδικές λέξεις διόρθωσης σφαλμάτων [4].



Εικόνα 2.1: Η δομή ενός QR κωδικού [5]

Υπάρχουν πολλοί τρόποι με τους οποίους μπορούν να δημιουργηθούν εύκολα QR κωδικοί. Ένας απλός τρόπος είναι από διάφορες εξειδικευμένες ιστοσελίδες όπου είναι δυνατή η εισαγωγή λέξεων,

URL, και εικόνων και αυτές μετατρέπονται απευθείας σε QR κωδικό. Συγκεκριμένα για την εφαρμογή του τεχνολογικού μουσείου του ΔΙΠΑΕ, χρησιμοποιήθηκε το QRServer API (από την πλατφόρμα goQR.me) για την δημιουργία των QR κωδικών που αντιστοιχούν στα εκθέματα του μουσείου. Η υπηρεσία επιστρέφει εικόνα τύπου PNG με τον αντίστοιχο κώδικα. Για παράδειγμα, ο σύνδεσμος https://api.qrserver.com/v1/create-qr-code/?size=150x150&data=Belcom_MC7000, επιστρέφει την εικόνα του QR για το έκθεμα Belcom MC7000. Παρομοίως και για τα υπόλοιπα εκθέματα δημιουργήθηκαν και οι υπόλοιποι σύνδεσμοι με το Microsoft Excel. Η συγκεκριμένη τεχνολογία αποτελεί μια εύκολη λύση για να ταυτοποιούνται τα εκθέματα με ταχύτητα και αξιοπιστία από την εφαρμογή. Στην παρακάτω εικόνα, φαίνεται το πως μπορούν να δημιουργηθούν εύκολα και γρήγορα QR κωδικοί.



Εικόνα 2.2: Παράδειγμα δημιουργίας QR κωδικού

2.2 Παρόμοιες εφαρμογές

Πολλά μουσεία στον κόσμο έχουν αξιοποιήσει ψηφιακές τεχνολογίες για να ενισχύσουν την εμπειρία των επισκεπτών τους. Η χρήση εφαρμογών για κινητές συσκευές, διαδραστικών χαρτών, ξεναγήσεων με επαυξημένη πραγματικότητα (AR) και προσωποποιημένων προτάσεων περιήγησης είναι πλέον ευρέως διαδεδομένες. Παρακάτω θα αναλυθούν ορισμένες από αυτές τις εφαρμογές.

2.2.1 Rijksmuseum, Amsterdam

Ένα χαρακτηριστικό παράδειγμα των όσων αναφέρθηκαν νωρίτερα, είναι η εφαρμογή του Rijksmuseum στο Άμστερνταμ. Μέσω της εφαρμογής αυτής, ο επισκέπτης δεν περιορίζεται στην παθητική θέαση των εκθεμάτων. Αντίθετα, μπορεί να επιλέξει συγκεκριμένες θεματικές διαδρομές, όπως για παράδειγμα «Η Χρυσή Εποχή» ή «Τα Αριστουργήματα του Ρέμπραντ», και να περιηγηθεί

στον χώρο με τη βοήθεια ενός έξυπνου χάρτη [6]. Η εφαρμογή προσφέρει πληροφορίες για τα έργα σε πολλαπλά επίπεδα, από βασικές λεζάντες μέχρι ιστορικές αναλύσεις και ηχητικές ξεναγήσεις. Επιπλέον, δίνει τη δυνατότητα στον χρήστη να αποθηκεύσει τα αγαπημένα του έργα και να δημιουργήσει τη δική του, προσωποποιημένη διαδρομή μέσα στο μουσείο, ενισχύοντας έτσι την αίσθηση εμπλοκής.

2.2.2 British Museum, London

Το Βρετανικό Μουσείο στο Λονδίνο έχει αναπτύξει μια ολοκληρωμένη εφαρμογή για κινητές συσκευές, με στόχο την ενίσχυση της εμπειρίας των επισκεπτών μέσω της ψηφιακής τεχνολογίας. Η εφαρμογή αυτή προσφέρει πρόσβαση σε περισσότερα από 250 σημαντικά αντικείμενα της συλλογής, με συνοδευτικά σχόλια από ειδικούς, καθώς και 65 εισαγωγές σε εκθέσεις οι οποίες είναι διαθέσιμες δωρεάν [7]. Οι χρήστες μπορούν να εξερευνήσουν το μουσείο μέσω αυτοκαθοδηγούμενων περιηγήσεων, να αποθηκεύσουν αγαπημένα αντικείμενα και να λάβουν πρακτικές πληροφορίες για την επίσκεψή τους. Η εφαρμογή υποστηρίζει εννέα γλώσσες, συμπεριλαμβανομένης της νοηματικής γλώσσας, και προσφέρει περιεχόμενο σε μορφή ήχου, βίντεο, κειμένου και εικόνων.

2.2.3 American Museum of Natural History, New York

Το Αμερικανικό Μουσείο Φυσικής Ιστορίας που βρίσκεται στη Νέα Υόρκη προσφέρει μια πολύ χρήσιμη εφαρμογή για τους επισκέπτες του, που ονομάζεται “Explorer”. Δεν πρόκειται απλώς για έναν συνηθισμένο χάρτη, αλλά για μια διαδραστική ψηφιακή πλατφόρμα. Μέσα από την εφαρμογή, ο επισκέπτης μπορεί να βρει πληροφορίες για τα εκθέματα, να περιηγηθεί με ευκολία στους χώρους του μουσείου και να δει υλικό, όπως βίντεο και σύντομες αφηγήσεις [8]. Ένα από τα πιο εντυπωσιακά χαρακτηριστικά της εφαρμογής είναι η χρήση της τεχνολογίας Bluetooth για εντοπισμό θέσης. Με αυτόν τον τρόπο, η εφαρμογή γνωρίζει πού βρίσκεται ο επισκέπτης μέσα στο μουσείο και του προτείνει σχετικά εκθέματα ή ενότητες που μπορεί να τον ενδιαφέρουν. Επίσης, υπάρχει δυνατότητα αλλαγής γλώσσας, κάτι που κάνει την εμπειρία πιο προσιτή σε όλους, ενώ η προσθήκη ήχου και εικόνας προσφέρει μια πιο ζωντανή και ενδιαφέρουσα ξενάγηση [9].

2.2.4 Museumfy

Ένα άλλο σημαντικό εργαλείο για μουσεία είναι η πλατφόρμα Museumfy. Πρόκειται για μία λύση η οποία δεν περιορίζεται σε ένα συγκεκριμένο μουσείο, αλλά προσφέρει υποδομές για πολιτιστικούς φορείς που θέλουν να αναπτύξουν τις δικές τους εφαρμογές ξενάγησης. Η Museumfy παρέχει τεχνικά εργαλεία για εισαγωγή περιεχομένου, σχεδιασμό διαδρομών, επιλογή γλωσσών και προσθήκη εικόνων, βίντεο ή ήχου [10].

Η διαφορά με άλλες, πιο αυστηρές πλατφόρμες, είναι ότι επιτρέπει προσαρμογή ανάλογα με τις δυνατότητες του μουσείου. Μπορεί να υποστηρίξει ένα απλό τοπικό μουσείο αλλά και μια μεγάλη πολιτιστική δομή. Αυτό την καθιστά χρήσιμο εργαλείο για εκείνα τα μουσεία που αναζητούν έναν πιο προσιτό τρόπο να αναβαθμίσουν την εμπειρία των επισκεπτών τους με τη χρήση τεχνολογίας, χωρίς να απαιτείται υψηλό κόστος ή εξειδικευμένο προσωπικό.

2.2.5 The Louvre Museum, Paris

Το Μουσείο του Λούβρου στο Παρίσι διαθέτει μια πλήρως ανεπτυγμένη εφαρμογή που στοχεύει στην υποστήριξη της εμπειρίας των επισκεπτών μέσα και έξω από τον μουσειακό χώρο. Η εφαρμογή περιλαμβάνει διαδραστικούς χάρτες, πληροφορίες για περισσότερα από 600 έργα τέχνης και

προτεινόμενες διαδρομές που βοηθούν τον χρήστη να οργανώσει την επίσκεψή του σύμφωνα με τα ενδιαφέροντά του. Υποστηρίζει περισσότερες από δώδεκα γλώσσες και προσφέρει δυνατότητα offline πλοήγησης, κάτι που την καθιστά ιδιαίτερα πρακτική για διεθνές κοινό [11]. Επιπλέον, η εφαρμογή παρέχει ιστορικά σχόλια και αναλύσεις που συνδέουν τα εκθέματα με το ευρύτερο πολιτιστικό τους πλαίσιο, ενισχύοντας τη μουσειακή αφήγηση και την κατανόηση των έργων.

2.2.6 Smithsonian Institution, Washington D.C.

Το Ινστιτούτο Smithsonian, στο οποίο υπάγονται δεκαεννέα μουσεία και γκαλερί σε ολόκληρες τις Ηνωμένες Πολιτείες, έχει δημιουργήσει μία ενιαία ψηφιακή εφαρμογή με την ονομασία Smithsonian Mobile. Η εφαρμογή αυτή συγκεντρώνει το περιεχόμενο όλων των πολιτιστικών φορέων του οργανισμού, προσφέροντας στους επισκέπτες τη δυνατότητα να αναζητούν πληροφορίες για εκθέματα, να πλοηγούνται στους χώρους των μουσείων μέσω GPS και να δημιουργούν τις δικές τους προσωπικές συλλογές με αντικείμενα που τους ενδιαφέρουν [12]. Επιπλέον, η εφαρμογή περιλαμβάνει ηχητικούς οδηγούς, πρακτικές πληροφορίες για κάθε χώρο, καθώς και άμεση πρόσβαση στο διαδικτυακό αποθετήριο του οργανισμού. Μέσα από αυτό, οι χρήστες μπορούν να εξερευνήσουν εκατοντάδες χιλιάδες τεκμήρια. Πρόκειται για ένα εργαλείο που ενισχύει ουσιαστικά τον εκπαιδευτικό ρόλο του Ινστιτούτου, προσφέροντας μία εμπειρία που δεν περιορίζεται μόνο στη διάρκεια της φυσικής επίσκεψης, αλλά συνεχίζεται και πέρα από αυτήν.

2.2.7 The Metropolitan Museum of Art (The Met), New York

Η εφαρμογή Met App αποτελεί το επίσημο ψηφιακό εργαλείο του Μητροπολιτικού Μουσείου Τέχνης της Νέας Υόρκης και έχει σχεδιαστεί ώστε να διευκολύνει και να εμπλουτίσει την εμπειρία του επισκέπτη. Μέσω της εφαρμογής, ο χρήστης μπορεί να περιηγηθεί στους χώρους του μουσείου, να ενημερωθεί για τις τρέχουσες εκθέσεις και να αποθηκεύσει αντικείμενα που του κεντρίζουν το ενδιαφέρον [13]. Επιπλέον, η εφαρμογή προσφέρει τη δυνατότητα δημιουργίας προσωποποιημένων διαδρομών, ανάλογα με τις προτιμήσεις του εκάστοτε επισκέπτη. Ιδιαίτερης σημασίας είναι και η παροχή ηχητικών και οπτικοακουστικών ξεναγήσεων για επιλεγμένα εκθέματα, κάτι που ενισχύει σημαντικά τη βιωματική διάσταση της επίσκεψης. Το περιβάλλον χρήσης είναι απλό, εύχρηστο και προσαρμόζεται τόσο σε επισκέπτες που βρίσκονται στο μουσείο όσο και σε όσους επιλέγουν να περιηγηθούν σε αυτό ψηφιακά.

2.2.8 Vatican Museums, Vatican City

Η επίσημη εφαρμογή των Μουσείων του Βατικανού έχει αναπτυχθεί με σκοπό να προσφέρει μια ολοκληρωμένη εμπειρία ξενάγησης, τόσο για επισκέπτες που βρίσκονται επί τόπου όσο και για όσους προτιμούν να περιηγηθούν εξ αποστάσεως.

Μέσα από την εφαρμογή, δίνεται η δυνατότητα για εικονικές επισκέψεις σε μερικά από τα πιο γνωστά σημεία του συγκροτήματος, όπως η Καπέλα Σιξτίνα και οι Πινακοθήκες, με παράλληλη υποστήριξη ηχητικών αφηγήσεων [14]. Επιπλέον, ο χρήστης μπορεί να δημιουργήσει τη δική του διαδρομή μέσα στο μουσείο, να χρησιμοποιήσει φίλτρα για να βρει συγκεκριμένα έργα τέχνης και να έχει πρόσβαση σε πληροφορίες που καλύπτουν τόσο την ιστορική όσο και τη θρησκευτική σημασία των εκθεμάτων. Η εφαρμογή διατίθεται σε πολλές γλώσσες, κάτι που καθιστά το περιεχόμενο πιο εύκολα προσβάσιμο από διεθνές κοινό και ενισχύει την εκπαιδευτική διάσταση των μουσείων.

Κεφάλαιο 3ο: Πλατφόρμες Υλοποίησης Εφαρμογών

3.1 Εισαγωγή

Τα τελευταία χρόνια, δημιουργούνται όλο και περισσότερες εφαρμογές για κινητά και όχι μόνο. Δεν είναι όμως τόσο απλό όσο να συνταχθεί ένα κομμάτι κώδικα. Πρέπει να χρησιμοποιηθούν εργαλεία που συνεργάζονται, να υπάρχει καλός σχεδιασμός και να υπολογιστούν πολλά τεχνικά θέματα. Οι χρήστες απαιτούν όλο και περισσότερες λειτουργίες και οι τεχνολογίες αλλάζουν συνέχεια, οπότε η επιλογή των εργαλείων είναι κάτι που παίζει σημαντικό ρόλο στην υλοποίηση μιας εφαρμογής. Μια εφαρμογή δεν γίνεται να τρέχει μόνο σε Android. Πρέπει να δουλεύει καλά και σε iOS ή web, για να καλύπτει όσο περισσότερο γίνεται, μεγαλύτερο εύρος χρηστών. Επίσης είναι βασικό να έχει καλή απόκριση, να είναι απλή στη χρήση και να μπορεί να επικοινωνεί με βάσεις δεδομένων ή cloud υπηρεσίες. Όταν μια εφαρμογή υλοποιείται με τη βοήθεια διαφόρων εργαλείων, είναι σημαντικό αυτά να μπορούν να συνεργαστούν χωρίς προβλήματα. Αν δεν υπάρχει καλή επικοινωνία μεταξύ τους, είναι σχεδόν βέβαιο πως κάπου θα παρουσιαστεί δυσλειτουργία. Στην περίπτωση της εφαρμογής για το τεχνολογικό μουσείο του ΔΙΠΑΕ, χρησιμοποιήθηκαν τρία βασικά εργαλεία: το Android Studio, το Flutter και το Supabase. Το Android Studio χρησιμοποιήθηκε για να συνταχθεί ο βασικός κώδικας της εφαρμογής, τόσο για Android συσκευές όσο και για την έκδοση που λειτουργεί εξ ίσου καλά και στο web. Το Flutter βοήθησε στο να φτιαχτεί το περιβάλλον που βλέπει ο χρήστης (UI), ενώ το Supabase κάλυψε την πλευρά του backend, δηλαδή την αποθήκευση των δεδομένων και τη σύνδεση με τη βάση. Ο τρόπος που χρησιμοποιήθηκαν όλα αυτά μαζί είχε σαν αποτέλεσμα μια εφαρμογή που λειτουργεί σωστά και μπορεί στο μέλλον να επεκταθεί αν χρειαστεί.

3.2 Android Studio

Το βασικό εργαλείο που χρησιμοποιήθηκε για την ανάπτυξη της εφαρμογής ήταν το Android Studio. Πρόκειται για το πρόγραμμα που προτείνει η ίδια η Google για δημιουργία εφαρμογών Android. Στηρίζεται πάνω στο IntelliJ IDEA, αλλά είναι προσαρμοσμένο για χρήση σε κινητές συσκευές [15]. Ένα από τα πιο χρήσιμα πράγματα που προσφέρει είναι ο εξομοιωτής, ο οποίος επιτρέπει να δοκιμάζεται ο κώδικας απευθείας, χωρίς να χρειάζεται να υπάρχει κινητό τηλέφωνο. Αυτό ήταν πολύ βολικό, ειδικά στην αρχή, που η εφαρμογή άλλαζε συνέχεια. Επίσης, τα templates που υπάρχουν στο πρόγραμμα βοήθησαν ώστε να στηθεί πιο εύκολα η βασική δομή της εφαρμογής [16].

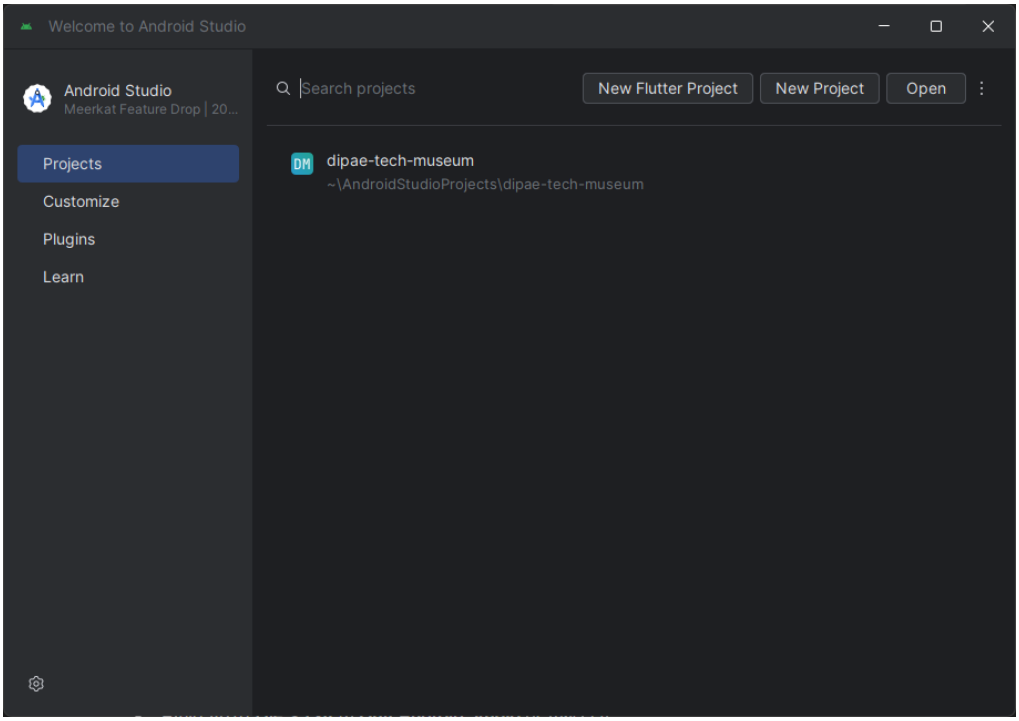
Το Android Studio υποστηρίζει δύο γλώσσες, την Java και την Kotlin. Έτσι, ο κάθε προγραμματιστής μπορεί να επιλέξει αυτή που του ταιριάζει ή αυτή που ζητάει το έργο. Ένα ακόμα χαρακτηριστικό που αποδείχθηκε χρήσιμο ήταν το live preview, δηλαδή η δυνατότητα να βλέπει κανείς σε πραγματικό χρόνο πώς φαίνονται οι αλλαγές που κάνει στο γραφικό περιβάλλον της εφαρμογής [17].

Το συγκεκριμένο εργαλείο κυκλοφόρησε πρώτη φορά το 2013 και σταδιακά πήρε τη θέση του Eclipse, που μέχρι τότε ήταν η πιο συχνή επιλογή για ανάπτυξη Android εφαρμογών. Πλέον, μπορεί να εγκατασταθεί σε Windows, macOS και Linux, και θεωρείται από τα πιο γνωστά προγράμματα για όσους ασχολούνται με Android [18].

3.2.1 Εγκατάσταση

Για να εγκατασταθεί το Android Studio, ο χρήστης πρέπει να επισκεφτεί την επίσημη ιστοσελίδα της Google [15], να επιλέξει την έκδοση που ταιριάζει στο λειτουργικό σύστημα του υπολογιστή του (Windows, macOS ή Linux) και να ακολουθήσει τα βήματα που δίνονται εκεί. Στην εγκατάσταση

περιλαμβάνονται και άλλα εργαλεία, όπως το Android SDK και ο εξομοιωτής, τα οποία είναι απαραίτητα για να λειτουργήσει σωστά το περιβάλλον ανάπτυξης.



Εικόνα 3.1: Αρχική οθόνη του Android Studio

Κατά την πρώτη εκκίνηση, το πρόγραμμα καθοδηγεί τον χρήστη μέσω ενός αυτόματου οδηγού εγκατάστασης (setup wizard), ο οποίος ρυθμίζει τα βασικά εργαλεία που απαιτούνται. Αυτά περιλαμβάνουν:

- Android SDK (Software Development Kit)
- AVD Manager (Android Virtual Device Manager)
- Build system (Gradle)
- Εργαλεία γραμμής εντολών (command-line tools)

Πριν ξεκινήσει η χρήση του εξομοιωτή, είναι απαραίτητο να έχει ενεργοποιηθεί η τεχνολογία Virtualization στο BIOS ή UEFI του υπολογιστή. Αυτή η ρύθμιση επιτρέπει τη σωστή λειτουργία του Android Emulator, που προσομοιώνει Android συσκευές στο περιβάλλον του υπολογιστή [16].

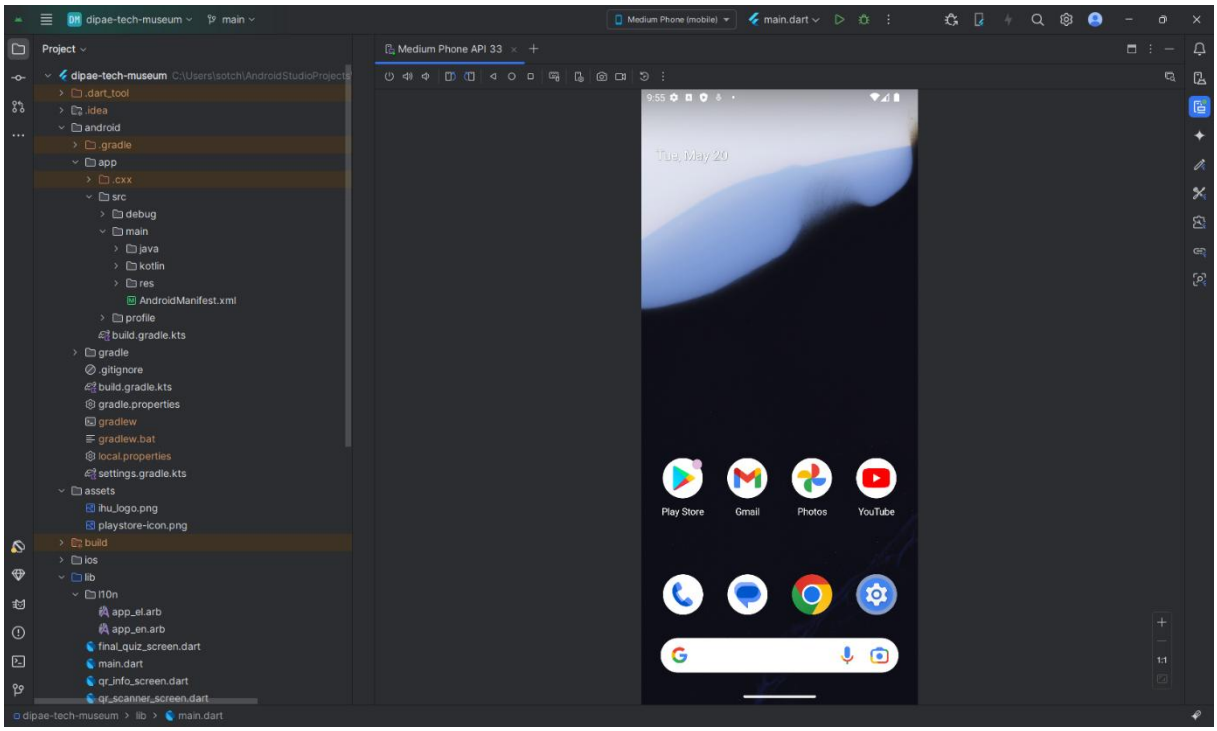
Επιπλέον, για την εύρυθμη λειτουργία του Android Studio απαιτείται και το Java Development Kit (JDK), επειδή πολλές δυνατότητες της πλατφόρμας βασίζονται στη γλώσσα Java. Συνιστάται η χρήση της έκδοσης Java 11 ή νεότερης, καθώς προσφέρει πλήρη συμβατότητα με τις πιο πρόσφατες εκδόσεις του Android Studio [15]. Σε αρκετές περιπτώσεις, το JDK περιλαμβάνεται ήδη στο αρχείο εγκατάστασης, αλλά ο χρήστης έχει και την επιλογή να το εγκαταστήσει ξεχωριστά, εφόσον το επιθυμεί.

3.2.2 Περιγραφή και Ανάλυση Λειτουργιών

Το Android Studio χρησιμοποιείται ευρέως για την ανάπτυξη εφαρμογών Android και ενσωματώνει αρκετές δυνατότητες που διευκολύνουν τον προγραμματιστή. Ένα από τα πιο βασικά εργαλεία είναι ο επεξεργαστής κώδικα, ο οποίος υποστηρίζει αυτόματη συμπλήρωση εντολών, εντοπισμό σφαλμάτων κατά την πληκτρολόγηση και άλλες λειτουργίες που κάνουν την ανάπτυξη πιο αποτελεσματική. Όταν ένα έργο περιλαμβάνει πολλά αρχεία και κλάσεις, αυτές οι δυνατότητες είναι ακόμη πιο χρήσιμες.

Μια άλλη σημαντική λειτουργία είναι το σύστημα Gradle, που βοηθά στην οργάνωση διαφορετικών εκδόσεων της εφαρμογής, όπως για δοκιμές ή τελική χρήση. Μέσα από αυτό, μπορούν επίσης να προστεθούν εξωτερικές βιβλιοθήκες που απαιτούνται για την ορθή λειτουργία της εφαρμογής [15].

Το Android Studio περιλαμβάνει επίσης έναν εξομοιωτή, γνωστό ως Android Emulator. Αυτό το εργαλείο επιτρέπει τη δημιουργία και χρήση εικονικών συσκευών με διάφορες προδιαγραφές, ώστε να μπορεί ο χρήστης να δοκιμάσει την εφαρμογή του χωρίς να χρειάζεται πραγματικό κινητό. Η δυνατότητα αυτή εξοικονομεί χρόνο και είναι ιδιαίτερα χρήσιμη όταν δεν υπάρχει πρόσβαση σε πολλές διαφορετικές συσκευές [16].



Εικόνα 3.2: Ο Emulator του Android Studio

Υπάρχει και το Android Profiler, ένα εργαλείο που δείχνει πόση μνήμη, CPU ή δικτύωση χρησιμοποιεί η εφαρμογή όσο τρέχει. Έτσι μπορεί να βρεθεί πιο εύκολα αν υπάρχει πρόβλημα απόδοσης ή αν η εφαρμογή χρησιμοποιεί αρκετούς πόρους του συστήματος, χωρίς λόγο.

Τέλος, στο Android Studio υπάρχει ενσωματωμένη υποστήριξη για Git. Ο χρήστης μπορεί να κάνει commit, pull, push και άλλες ενέργειες για να διαχειρίζεται τον πηγαίο του κώδικα, χωρίς να χρειάζεται εξωτερικά προγράμματα. Αυτό βοηθάει και όταν δουλεύουν δύο ή περισσότερα άτομα πάνω στο ίδιο project.

3.2.3 Πλεονεκτήματα και Μειονεκτήματα

Το Android Studio έχει αρκετά πλεονεκτήματα που το κάνουν ιδιαίτερα χρήσιμο για όσους ασχολούνται με την ανάπτυξη εφαρμογών Android. Το πιο σημαντικό είναι ότι είναι το επίσημο IDE που υποστηρίζεται από τη Google. Αυτό σημαίνει ότι ενημερώνεται συχνά και είναι πάντα συμβατό με τις πιο πρόσφατες εκδόσεις του Android. Έτσι, κάποιος που το χρησιμοποιεί ξέρει ότι δουλεύει σε ένα περιβάλλον σταθερό, ασφαλές και σύγχρονο [15].

Ένα άλλο βασικό πλεονέκτημα είναι ότι μέσα στο Android Studio υπάρχουν συγκεντρωμένα όλα τα εργαλεία που χρειάζονται. Ο χρήστης δεν χρειάζεται να εγκαταστήσει εξωτερικά προγράμματα ή πρόσθετα για να ξεκινήσει. Όλα, από τη συγγραφή του κώδικα μέχρι τη δοκιμή, το debugging και την τελική κατασκευή του αρχείου της εφαρμογής, γίνονται από το ίδιο σημείο. Αυτό εξοικονομεί χρόνο και κάνει την εργασία πιο άμεση.

Το Android Studio δίνει τη δυνατότητα να προστεθούν πρόσθετα (plugins), έτσι ώστε το περιβάλλον να προσαρμόζεται ανάλογα με το τι χρειάζεται κάθε έργο. Για παράδειγμα, μπορεί να εγκαταστήσει κανείς υποστήριξη για Flutter, για βάσεις δεδομένων ή για έλεγχο του κώδικα. Αυτό το κάνει πιο ευέλικτο και βοηθάει στο να δουλεύει ο καθένας όπως τον βολεύει.

Από την άλλη, υπάρχουν και κάποια πράγματα που μπορεί να δημιουργήσουν δυσκολίες. Το πρόγραμμα χρειάζεται αρκετούς πόρους για να τρέξει ομαλά. Αν για παράδειγμα λειτουργεί μαζί με τον εξομοιωτή, τότε ο υπολογιστής μπορεί να κολλάει. Αυτό γίνεται πιο έντονο σε παλιά μηχανήματα ή όταν η μνήμη RAM είναι κάτω από 8 GB [17].

Ένα άλλο θέμα είναι η πολυπλοκότητα του Android Studio. Αν κάποιος δεν έχει ξαναδουλέψει με IDE ή με Android περιβάλλοντα, στην αρχή θα του φανεί βαρύ. Υπάρχουν πολλές έννοιες, όπως τα SDKs, το Gradle και τα AVDs, που χρειάζονται χρόνο μέχρι να τα μάθει κανείς. Για αυτό, δεν είναι ιδανικό για τελείως αρχάριους, γιατί η καμπύλη εκμάθησης είναι κάπως απότομη.

3.2.4 Παρόμοιες Πλατφόρμες

Αν και το Android Studio είναι το βασικό εργαλείο για Android, υπάρχουν και εναλλακτικές. Η πιο κοντινή είναι η IntelliJ IDEA. Στην ουσία είναι η βάση πάνω στην οποία έχει χτιστεί το Android Studio, και μπορεί να γίνει χρήση της και για Android εφαρμογές αν εγκατασταθεί plugin. Κάποιοι τη βρίσκουν πιο “ελαφριά” και πιο απλή.

Μια άλλη επιλογή είναι το Visual Studio από τη Microsoft, που σε συνδυασμό με το Xamarin, δίνει τη δυνατότητα για ανάπτυξη εφαρμογών σε Android, iOS και Windows με κοινό κώδικα σε C#. Είναι ιδανικό για όσους έχουν εμπειρία σε .NET και θέλουν να καλύψουν πολλά λειτουργικά χωρίς να γράφουν τα ίδια πράγματα από την αρχή.

Υπάρχουν και άλλες λύσεις όπως το Cordova ή το Ionic, που βασίζονται σε τεχνολογίες ιστού (HTML, CSS, JavaScript). Είναι καλή επιλογή για απλές εφαρμογές, κυρίως για όσους προέρχονται από web development. Παρόλα αυτά, οι εφαρμογές τους δεν έχουν την ίδια απόδοση με τις native Android εφαρμογές.

Αν και οι επιλογές είναι αρκετές, το Android Studio παραμένει η πιο αξιόπιστη πρόταση, ειδικά για πιο σύνθετα ή επαγγελματικά projects, λόγω της σύνδεσής του με το Android SDK και της στήριξης από τη Google.

3.3 Flutter

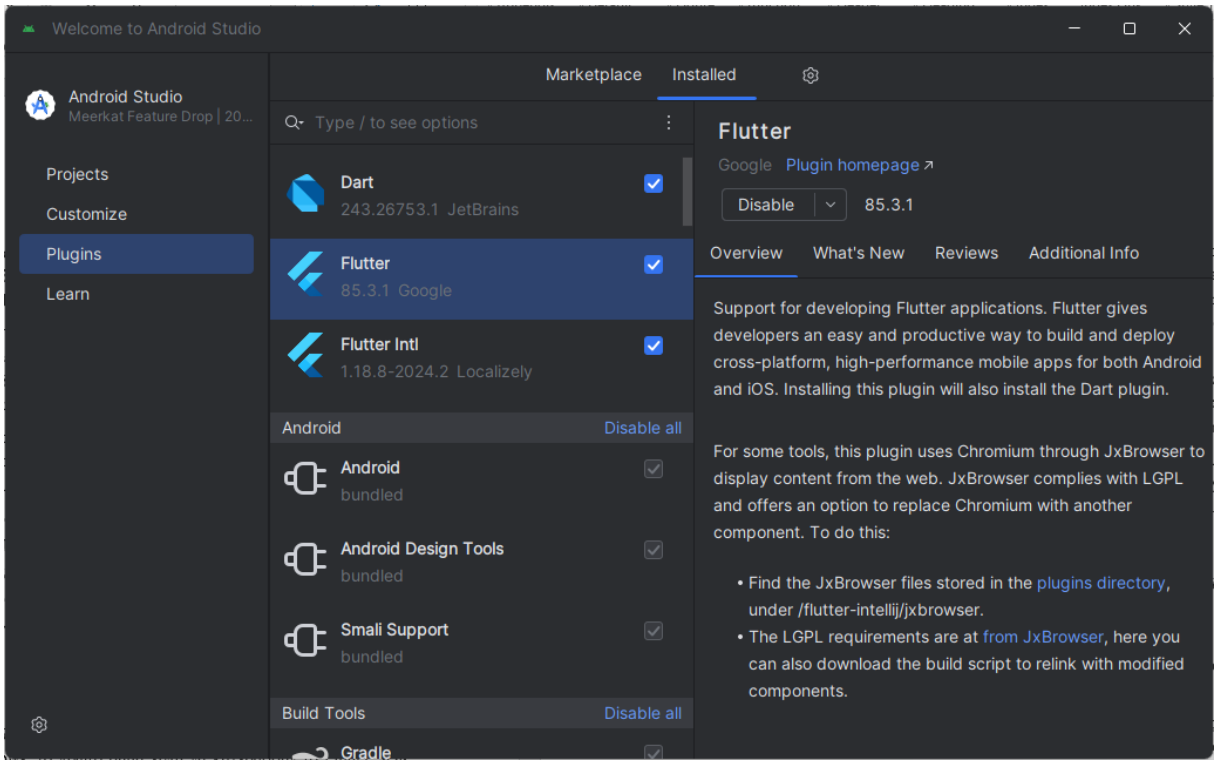
Το Flutter είναι ένα framework που δημιουργήθηκε από την Google και βοηθάει στην ανάπτυξη εφαρμογών για πολλές πλατφόρμες. Δηλαδή, μπορεί να συνταχθεί ένας κώδικας και να λειτουργεί με τον ίδιο τρόπο σε κινητό, υπολογιστή ή ακόμα και στο web, χωρίς να χρειάζεται να δημιουργηθεί ξεχωριστά για κάθε σύστημα [19]. Αυτό είναι πρακτικό και εξοικονομεί πολύ χρόνο, ειδικά σε μεγαλύτερα έργα.

Η γλώσσα που χρησιμοποιεί το Flutter λέγεται Dart. Δεν είναι τόσο γνωστή όσο η Java ή η Python, αλλά έχει καλά χαρακτηριστικά, όπως ότι μεταγλωττίζεται απευθείας σε native κώδικα. Αυτό σημαίνει ότι η εφαρμογή δεν καθυστερεί και λειτουργεί πιο ομαλά [20].

Ένα πολύ χρήσιμο χαρακτηριστικό είναι το hot reload. Με αυτό, όταν κάποιος αλλάζει τον κώδικα, μπορεί να δει την αλλαγή στην εφαρμογή κατευθείαν, χωρίς να την εκτελέσει απ' την αρχή. Είναι κάτι που βοηθά πολύ στον σχεδιασμό του UI και γενικά κάνει πιο γρήγορη την ανάπτυξη [21].

Επίσης, έχει πολλά έτοιμα widgets που κάνουν πιο εύκολη τη δημιουργία του περιβάλλοντος χρήστη. Δεν χρειάζεται να ξεκινάει κανείς από το μηδέν , τα περισσότερα στοιχεία υπάρχουν ήδη [22].

Τέλος, το Flutter έχει δική του μηχανή σχεδίασης, που λέγεται Skia. Αυτό σημαίνει ότι το UI φαίνεται ίδιο σε Android και iOS, χωρίς μεγάλες διαφορές. Δεν χρησιμοποιεί τα έτοιμα συστήματα κάθε πλατφόρμας, αλλά τα σχεδιάζει όλα από την αρχή [23].



Εικόνα 3.3: το Flutter plugin όπως φαίνεται απο το Android Studio

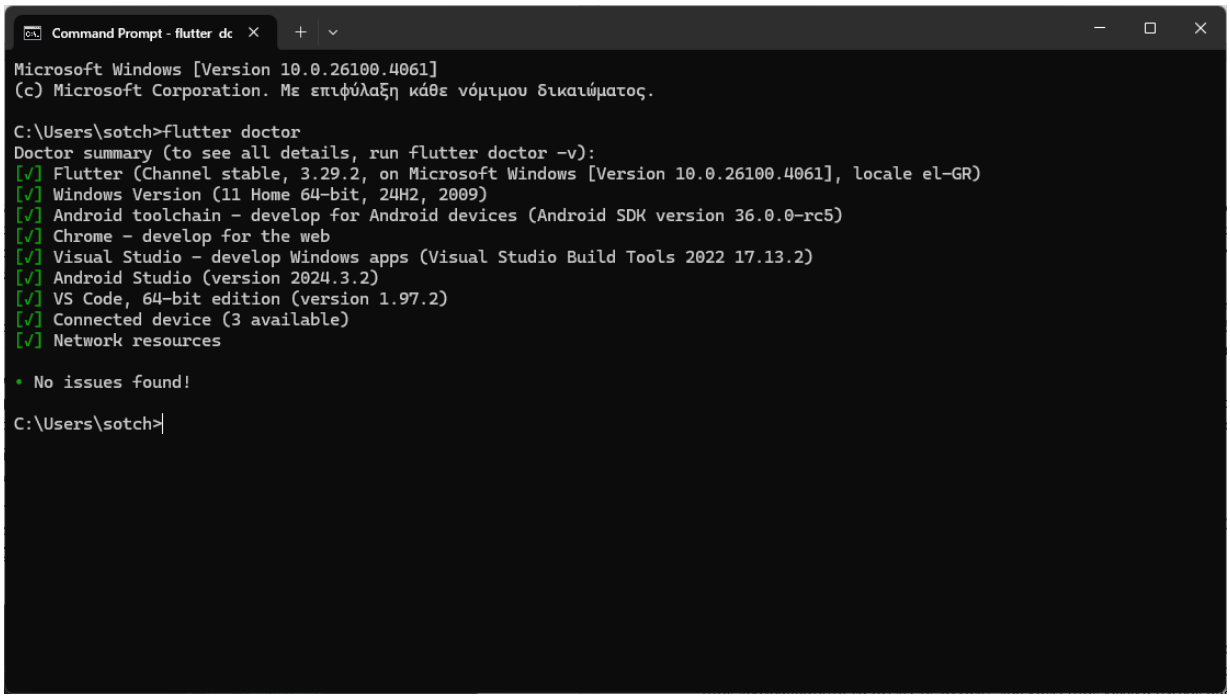
3.3.1 Εγκατάσταση

Η εγκατάσταση του Flutter είναι σχετικά απλή, αλλά αλλάζει λίγο ανάλογα με το λειτουργικό σύστημα. Για όσους χρησιμοποιούν Windows, το πρώτο βήμα είναι να κατεβάσουν το Flutter SDK από την επίσημη ιστοσελίδα. Μετά γίνεται αποσυμπίεση του αρχείου zip σε έναν φάκελο της επιλογής τους, π.χ. στην επιφάνεια εργασίας ή στο C:\Flutter.

Ένα σημαντικό βήμα μετά από αυτό είναι να προστεθεί η διαδρομή του φακέλου flutter/bin στο περιβάλλον του συστήματος (PATH). Αυτό γίνεται για να μπορεί να αναγνωρίζεται η εντολή flutter από τη γραμμή εντολών (cmd ή terminal). Αν αυτό δεν γίνει, το Flutter δε θα λειτουργεί σωστά από το command line.

Στη συνέχεια, πρέπει να εγκατασταθεί κάποιο IDE, όπως το Android Studio ή το Visual Studio Code. Και στα δύο, είναι απαραίτητο να μπουν και τα plugins για το Flutter και τη γλώσσα Dart. Τα plugins προσθέτουν επιλογές και εργαλεία ώστε να μπορεί να δημιουργείται, να τρέχει και να γίνεται debug μια εφαρμογή Flutter κανονικά.

Τέλος, υπάρχει η εντολή flutter doctor. Αυτή είναι ένα πολύ χρήσιμο εργαλείο γιατί ελέγχει αν έχει στηθεί σωστά το περιβάλλον, αν λείπει κάτι, και προτείνει τι να γίνει για να φτιαχτεί. Π.χ. αν λείπει το Android SDK ή το emulator, το εμφανίζει εκεί και δίνει οδηγίες [24][25].



Εικόνα 3.4: Παράδειγμα εκτέλεσης της εντολής flutter doctor από τερματικό Windows

3.3.2 Περιγραφή και Ανάλυση Λειτουργιών

Το Flutter διαθέτει αρκετές λειτουργίες που το κάνουν ιδιαίτερα χρήσιμο σε όσους φτιάχνουν εφαρμογές. Ένα από τα πιο γνωστά χαρακτηριστικά είναι το hot reload. Αυτό δίνει τη δυνατότητα στον προγραμματιστή να βλέπει άμεσα τις αλλαγές που κάνει στον κώδικα, χωρίς να χρειάζεται να επανεκκινεί την εφαρμογή κάθε φορά. Είναι πολύ βοηθητικό ειδικά όταν σχεδιάζεται το περιβάλλον χρήστη (UI), αφού επιτρέπει δοκιμές και διορθώσεις σε πραγματικό χρόνο [21].

Ένα ακόμα δυνατό στοιχείο του Flutter είναι τα έτοιμα widgets, δηλαδή προκατασκευασμένα κομμάτια του UI όπως κουμπιά, φόρμες, λίστες κ.ά. Αυτά μπορούν να προσαρμοστούν ανάλογα με τις ανάγκες κάθε εφαρμογής, και γλιτώνουν αρκετή δουλειά γιατί δεν χρειάζεται να φτιαχτούν από την αρχή [22].

Το Flutter χρησιμοποιεί τη γλώσσα Dart, η οποία μεταγλωττίζεται απευθείας σε εγγενή κώδικα (native code). Αυτό σημαίνει ότι η εφαρμογή τρέχει απευθείας στο λειτουργικό, χωρίς ενδιάμεσους μεταφραστές, κάτι που βοηθά στην ταχύτητα και τη συνολική απόδοση [20].

Ένα άλλο πλεονέκτημα είναι ότι ο ίδιος κώδικας μπορεί να χρησιμοποιηθεί για περισσότερες από μία πλατφόρμες. Δηλαδή, με μια βάση κώδικα μπορεί να χτιστεί εφαρμογή για Android, iOS, υπολογιστή και web. Έτσι, μειώνεται και ο χρόνος που απαιτείται και το συνολικό κόστος [19].

3.3.3 Πλεονεκτήματα και Μειονεκτήματα

Το Flutter έχει αρκετά θετικά στοιχεία, αν και όπως κάθε εργαλείο, δεν είναι ιδανικό σε όλα. Ένα βασικό θετικό είναι η ταχύτητα. Με το hot reload, ό,τι αλλαγές γίνονται στον κώδικα εμφανίζονται σχεδόν αμέσως στην εφαρμογή. Έτσι, ο χρόνος δοκιμών και διορθώσεων μειώνεται σημαντικά [21].

Επίσης, δεν είναι ανάγκη να φτιάχνει κάποιος ξεχωριστή εφαρμογή για κάθε πλατφόρμα. Με τον ίδιο κώδικα, μπορεί να καλύψει Android, iOS και desktop, κάτι που εξοικονομεί χρόνο και προσπάθεια [19].

Ένα ακόμη πλεονέκτημα είναι η ύπαρξη πολλών έτοιμων widgets. Αυτά είναι έτοιμα κομμάτια UI που μπορούν να χρησιμοποιηθούν χωρίς να χρειάζεται να σχεδιαστούν από την αρχή, κάτι που διευκολύνει πολύ τη διαδικασία [22].

Από την άλλη πλευρά, η γλώσσα Dart που χρησιμοποιείται στο Flutter δεν είναι ιδιαίτερα διαδεδομένη. Αυτό σημαίνει ότι μπορεί να μπερδευτεί κάποιος που δεν την έχει δουλέψει ξανά, αν και το συντακτικό της θυμίζει άλλες γλώσσες όπως Java ή Kotlin [20].

Τέλος, αν και το Flutter έχει εξελιχθεί πολύ τα τελευταία χρόνια, κάποιες βιβλιοθήκες τρίτων δεν είναι ακόμη τόσο πλήρεις ή σταθερές όσο αυτές που υπάρχουν σε άλλες πιο παλιές πλατφόρμες.

3.3.4 Παρόμοιες Πλατφόρμες

Δεν ήταν ποτέ το Flutter η μοναδική λύση για ανάπτυξη εφαρμογών για πολλές πλατφόρμες. Υπάρχουν κι άλλες λύσεις, παρόμοιες με το Flutter. Η λογική τους κοινή: ένας κώδικας, πολλά περιβάλλοντα. Μια προσέγγιση που επιδιώκει να διευκολύνει το έργο των προγραμματιστών και να μειώσει το χρόνο περάτωσης ενός project.

Το React Native είναι ίσως αυτό που συναντά κανείς πιο συχνά. Ανήκει στη Meta, χρησιμοποιεί JavaScript, και γύρω του έχει αναπτυχθεί μια αρκετά μεγάλη κοινότητα. Διαθέτει πολλά έτοιμα εργαλεία, χωρίς να πιέζει. Όταν προκύπτει ανάγκη για κάτι πιο σύνθετο, είναι εκεί [26].

Το Xamarin λειτουργεί διαφορετικά. Βασίζεται στη C# και απευθύνεται σε όσους ήδη κινούνται στον χώρο του Visual Studio. Δεν επιβάλλεται, για κάποιους είναι απλώς πιο γνώριμο. Και αυτή η αίσθηση οικειότητας έχει τη δική της βαρύτητα.

Το Ionic στέκεται πιο κοντά στον ιστό. HTML, CSS, JavaScript, τεχνολογίες που έχουν μακρά παρουσία. Αν η απαίτηση δεν είναι το μέγιστο της εγγενούς απόδοσης, αλλά κάτι λειτουργικό που "να τρέχει", τότε αυτή η επιλογή επαρκεί.

Το NativeScript, τέλος, δεν εμφανίζεται συχνά, αλλά σε ορισμένες περιπτώσεις είναι αυτό που ταιριάζει. Χρησιμοποιεί JavaScript ή TypeScript και δίνει πρόσβαση σε native APIs χωρίς περιττή πολυπλοκότητα. Δεν προσφέρει πολλά, μόνο όσα χρειάζονται.

3.3.5 Νέες Δυνατότητες και Βελτιώσεις στην Έκδοση Flutter 3.24

Με την έκδοση 3.24, το Flutter ενημερώνεται διακριτικά. Όχι με εντυπωσιακές κινήσεις, αλλά με μικρές, σταθερές βελτιώσεις. Δεν είναι οι αλλαγές που τραβούν το βλέμμα. Είναι εκείνες που, κατά τη διάρκεια της δουλειάς, γίνονται αισθητές. Πιο ομαλό scrolling, πιο σίγουρη απόδοση, πιο φυσική αίσθηση.

3.3.6 Απόδοση GPU και Rendering με Impeller

Η μετάβαση στο Impeller δεν έγινε για να παρουσιαστεί κάτι καινούριο. Έγινε επειδή χρειαζόταν. Πρόκειται για έναν rendering πυρήνα πιο κοντά στο υλικό και πιο συνεπή. Το αποτέλεσμα είναι γραφικά που εμφανίζονται πιο καθαρά στην οθόνη. Σκηνές σε 2D ή 3D, shaders και όλα αυτά που παλιότερα ήθελαν προσπάθεια, πλέον μοιάζουν πιο προσιτά.

Η υποστήριξη παραμένει πειραματική, αλλά ήδη υπάρχει σε Android, iOS και macOS. Ο σχεδιασμός προβλέπει την επέκτασή της σε όλες τις υποστηριζόμενες πλατφόρμες [23].

3.3.7 Υποστήριξη Πολλαπλών Προβολών στον Ιστό

Μέχρι πρότινος, οι web εφαρμογές με Flutter περιορίζονταν σε μία προβολή. Η έκδοση 3.24 αλλάζει αυτό το δεδομένο. Πλέον, είναι δυνατή η προσθήκη πολλαπλών HTML προβολών, δηλαδή ανεξάρτητα σημεία Flutter σε μια ιστοσελίδα [19].

Η προσθήκη δεν είναι θεαματική. Όμως προσφέρει ευελιξία. Γιατί μια εφαρμογή δεν είναι πάντα μία κάτι ενιαίο. Κάποιες φορές είναι πολλά κομμάτια που συνυπάρχουν.

3.3.8 Νέα Widgets και UI Βελτιώσεις

Κάθε νέο widget μοιάζει με εργαλείο που προστίθεται αθόρυβα. Δεν ξεχωρίζει, δεν επιβάλλεται. Απλώς υπάρχει, για τη στιγμή που θα χρειαστεί. Slivers με μεταβλητό μέγεθος, κεφαλίδες που παραμένουν στη θέση τους κατά την κύλιση. Ένα CarouselView, για περιεχόμενο που αλλάζει απαλά και το TreeView, για περιπτώσεις με πολυεπίπεδες δομές [22].

Καμία από αυτές τις προσθήκες δεν αναζητά εντυπώσεις. Δημιουργήθηκαν επειδή υπήρξε ανάγκη και αυτό αρκεί. [22].

3.3.9 Βελτιωμένη Επιλογή Κειμένου (SelectionArea)

Το SelectionArea δεν έγινε πιο έξυπνο για να θεωρηθεί εξελιγμένο. Έγινε απλώς πιο κοντά σε αυτό που περιμένει ο χρήστης. Διπλό πάτημα, τριπλό πάτημα, επιλογή λέξης, επιλογή παραγράφου. Κινήσεις απλές, φυσικές. Αυτές που δεν χρειάζονται σκέψη — απλώς γίνονται [22].

3.3.10 Νέες Εκδόσεις του SharedPreferences

Η ανάγκη υπήρχε εδώ και καιρό: η αποθήκευση να γίνεται χωρίς να παγώνει η διεπαφή. Η νέα έκδοση προσφέρει δύο επιλογές. Το SharedPreferencesAsync μεταφέρει την επεξεργασία στο παρασκήνιο, ήσυχα. Το SharedPreferencesWithCache κρατά τα δεδομένα πιο κοντά, επιταχύνοντας την πρόσβαση. Και στις δύο περιπτώσεις, το αποτέλεσμα είναι απλότητα στη χρήση και λιγότερη αναμονή [19].

3.3.11 Βελτίωση στην Απόδοση Εικόνων

Η αλλαγή εδώ είναι μικρή, αλλά ουσιαστική. Η προεπιλεγμένη ποιότητα φιλτραρίσματος δεν είναι πλέον στο χαμηλό και το αποτέλεσμα είναι εικόνες που παραμένουν καθαρές, ακόμα και σε μικρότερες διαστάσεις χωρίς να επηρεάζεται η απόδοση. Γενικά υπάρχει μια καλύτερη ισορροπία [23].

3.3.12 Ενσωμάτωση με Dart 3.5

Η Dart ακολούθησε κι αυτή μια πορεία σταθερής εξέλιξης. Η έκδοση 3.5 εισάγει pattern matching, record types, macros. Όχι για να φανούν οι δυνατότητες, αλλά για να γίνει ο κώδικας πιο εκφραστικός και να συντηρείται πολύ πιο εύκολα. Να διατηρείται πιο εύκολα. Όταν η γλώσσα πλησιάζει τον τρόπο σκέψης, όλα απλοποιούνται [20].

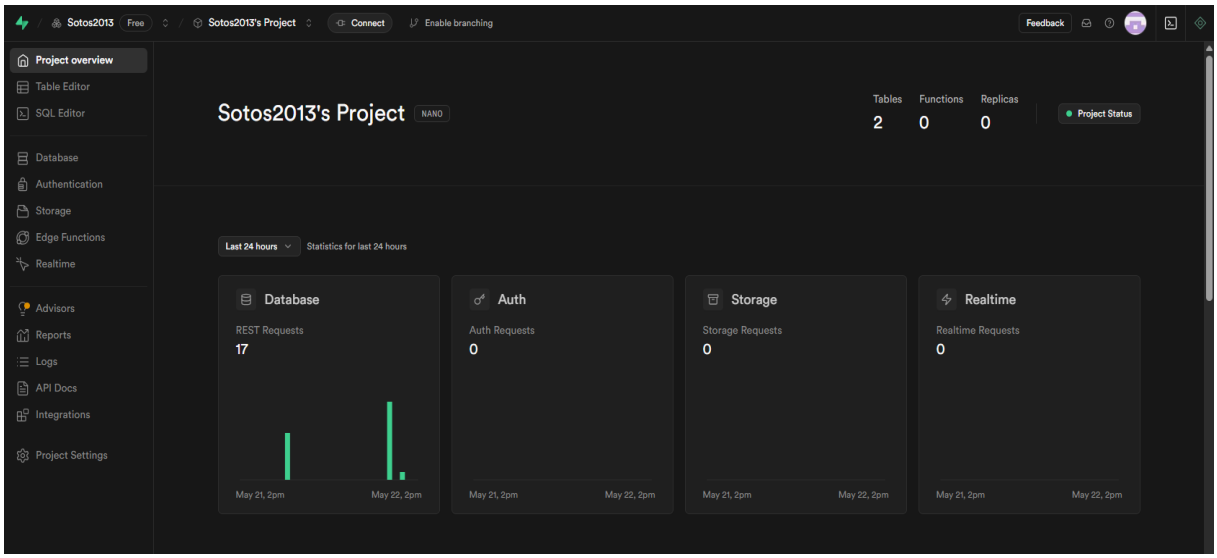
3.3.13 Βελτιώσεις στα DevTools

Τέλος, τα DevTools ενισχύθηκαν, όχι με νέες φανταχτερές λειτουργίες, αλλά με καλύτερα μέσα παρατήρησης. Δίνουν μια καθαρότερη εικόνα της εφαρμογής, εντοπίζοντας καθυστερήσεις, υπερβάσεις πόρων. Δεν προσθέτουν πολυπλοκότητα. Απλώς βοηθούν να γίνει η δουλειά πιο συνειδητά [19].

3.4 Supabase

Το Supabase είναι ένα εργαλείο που μπορεί να βοηθήσει κάποιον να φτιάξει την υποδομή πίσω από μια εφαρμογή χωρίς να ασχοληθεί με server ή με περίπλοκα backend συστήματα. Στην ουσία, είναι μια βάση δεδομένων (PostgreSQL) που έρχεται με πολλά έτοιμα πράγματα γύρω της, όπως σύστημα αυθεντικοποίησης χρηστών, δυνατότητα να βλέπει κανείς σε πραγματικό χρόνο τι αλλάζει στα δεδομένα και ένας χώρος για αρχεία [27].

Μόλις δημιουργηθεί ένας πίνακας στη βάση, το Supabase δίνει αυτόματα ένα τρόπο να διαβάζονται και να στέλνονται δεδομένα από και προς αυτόν τον πίνακα, χωρίς να χρειαστεί επιπλέον backend κώδικας. Αυτό γίνεται με ένα σύστημα που λέγεται PostgREST. Το API δημιουργείται μόνο του, αρκεί να υπάρχει ο πίνακας. Έτσι, ο προγραμματιστής επικεντρώνεται στο πώς θα φτιάξει την εφαρμογή και όχι στο πώς θα επικοινωνεί με τον server [27].



Εικόνα 3.5: Η αρχική σελίδα ενός project στο Supabase

Το θέμα της ασφάλειας καλύπτεται με κανόνες RLS (Row Level Security), που ουσιαστικά επιτρέπουν ή απαγορεύουν την πρόσβαση σε δεδομένα, ανάλογα με τον χρήστη. Αυτοί οι κανόνες δεν είναι πολύ δύσκολοι, αλλά χρειάζονται κάποια προσοχή. Όλα συνδέονται και με το σύστημα αυθεντικοποίησης που έχει το Supabase, το οποίο υποστηρίζει login με email, κωδικό, ακόμα και με OAuth, αν χρειάζεται [27].

Αξιοσημείωτο είναι και το real-time κομμάτι. Όταν κάποια δεδομένα αλλάξουν, μπορεί η εφαρμογή να ενημερωθεί αυτόματα. Αυτό είναι χρήσιμο όταν υπάρχει ανάγκη να βλέπει κάποιος τι συμβαίνει «ζωντανά» χωρίς να κάνει refresh. Το χρησιμοποιούν για παράδειγμα σε chats ή σε dashboards [27].

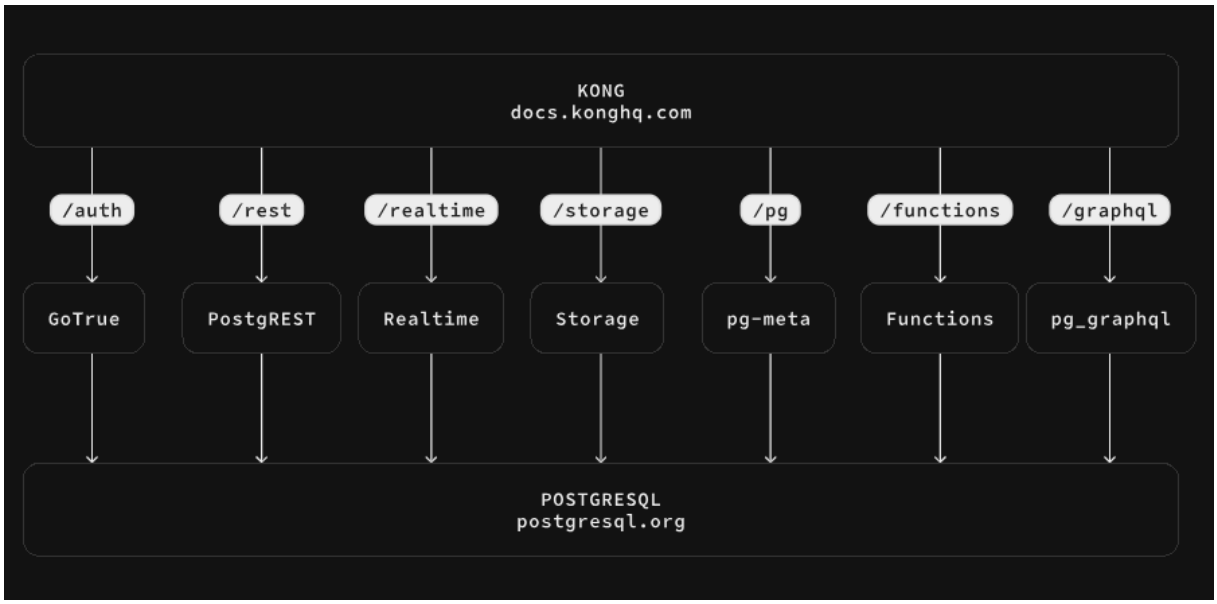
Τέλος, υπάρχει και σύστημα αποθήκευσης αρχείων. Οι εικόνες ή τα έγγραφα μπορούν να μπουν σε «φακέλους» (buckets) και να συνδεθούν με χρήστες. Ο κάθε χρήστης μπορεί να έχει δικαιώματα μόνο στα δικά του αρχεία. Όλα αυτά λειτουργούν καλά μαζί, γιατί είναι φτιαγμένα να συνεργάζονται. Δεν χρειάζεται ξεχωριστή πλατφόρμα για τα δεδομένα και άλλη για τις εικόνες — όλα είναι στο ίδιο περιβάλλον [27].

3.4.1 Αρχιτεκτονική και Υποδομή

Το Supabase, στο πιο βασικό του επίπεδο, βασίζεται στην PostgreSQL. Πρόκειται για μια πολύ γνωστή σχεσιακή βάση δεδομένων, που χρησιμοποιείται ευρέως και θεωρείται από τις πιο αξιόπιστες. Αυτό που κάνει το Supabase να ξεχωρίζει είναι ότι δεν χρειάζεται να δημιουργήσει κάποιος REST API από το μηδέν , το API “χτίζεται” αυτόματα πάνω στην PostgreSQL. Δηλαδή, κάθε φορά που προστίθεται ένας πίνακας στη βάση, δημιουργείται αντίστοιχο endpoint. Αυτό μειώνει αρκετά την ανάγκη για custom backend [28].

Το Supabase, πέρα από τα βασικά του, έχει και μερικά επιπλέον εργαλεία που συνεργάζονται μεταξύ τους. Ένα από αυτά είναι το PostgREST — και αυτό είναι που υλοποιεί τα endpoints πάνω στη βάση, χωρίς να χρειάζεται ο προγραμματιστής να δημιουργήσει backend για κάθε λειτουργία.

Ένα άλλο χρήσιμο στοιχείο είναι το Realtime. Με αυτό, μπορεί να παρακολουθεί κανείς αλλαγές στη βάση τη στιγμή που συμβαίνουν. Χωρίς refresh, χωρίς καθυστερήσεις — ιδανικό για εφαρμογές τύπου chat ή για dashboards που δείχνουν ενημερωμένα δεδομένα σε πραγματικό χρόνο [28].



Εικόνα 3.6: Παράδειγμα αρχιτεκτονικού διαγράμματος Supabase [28]

Όμως, τα πράγματα δεν είναι εντελώς αυτοματοποιημένα. Ο χρήστης (ή μάλλον ο διαχειριστής του συστήματος) πρέπει να ρυθμίσει σωστά κάποιες παραμέτρους. Ένα από τα πιο σημαντικά σημεία είναι τα Row Level Security (RLS) policies. Πρόκειται για φίλτρα — και αν δεν ενεργοποιηθούν σωστά, μπορεί κάποιος χρήστης να βλέπει δεδομένα που δεν θα έπρεπε. Είναι, δηλαδή, κάτι που πρέπει να προσεχθεί.

Ακόμα ένα στοιχείο που αξίζει να σημειωθεί είναι ότι η πλατφόρμα δεν περιορίζεται μόνο στο cloud. Υπάρχει η δυνατότητα να εγκατασταθεί τοπικά, σε δικό του server. Αυτό μπορεί να φανεί χρήσιμο σε ομάδες που δεν θέλουν να εξαρτώνται από εξωτερικούς παρόχους, ή σε περιβάλλοντα που απαιτούν μεγαλύτερο έλεγχο στα δεδομένα — για παράδειγμα, πανεπιστήμια ή ιδιωτικούς οργανισμούς.

3.4.2 Αυθεντικοποίηση και Διαχείριση Χρηστών

Ένα από τα πρώτα πράγματα που χρειάζεται κάθε εφαρμογή είναι κάποια μορφή σύνδεσης ή εγγραφής χρηστών. Το Supabase προσφέρει αυτή τη δυνατότητα από την αρχή, χωρίς ο χρήστης να χρειάζεται να υλοποίησει κάτι από την αρχή. Μέσα από την υπηρεσία που λέγεται GoTrue, μπορεί κάποιος να προσθέσει login με email και κωδικό, και με λίγες ρυθμίσεις να ενεργοποιήσει social logins, όπως με Google ή GitHub [27].

Το ωραίο είναι ότι, στην πράξη, δεν χρειάζεται να γράψει κάποιος πολύπλοκο κώδικα. Οι βασικές εντολές υπάρχουν ήδη στην επίσημη βιβλιοθήκη, και το μόνο που έχει να κάνει ο προγραμματιστής είναι να τις «καλέσει» μέσα στην εφαρμογή του. Αυτό κάνει το Supabase αρκετά προσβάσιμο, ακόμα και για όσους δεν έχουν εμπειρία με authentication flows.

Επίσης, υπάρχουν και επιλογές για ανάκτηση κωδικού ή επαλήθευση email, που είναι έτοιμες out-of-the-box. Όλα αυτά γίνονται με την αποστολή emails μέσω μιας υπηρεσίας που ορίζεις (π.χ. SMTP). Αν κάποιος θέλει περισσότερη ασφάλεια, μπορεί να βάλει κανόνες πρόσβασης, για παράδειγμα ώστε οι χρήστες να βλέπουν μόνο τα δικά τους δεδομένα, χρησιμοποιώντας τα λεγόμενα Row Level Security (RLS).

3.4.3 Ενσωμάτωση με Εφαρμογές Flutter

Όταν δουλεύει κανείς με Flutter και χρειάζεται να στηθούν backend λειτουργίες σχετικά γρήγορα, το Supabase μπορεί να φανεί ιδιαίτερα χρήσιμο. Δεν πρόκειται για κάτι που απαιτεί βαθιά παραμετροποίηση. Υπάρχει η βιβλιοθήκη supabase_flutter, η οποία προστίθεται απλά στο pubspec.yaml, όπως κάθε άλλη. Από εκεί κι έπειτα, τα περισσότερα βασικά όπως εγγραφή χρηστών, αποθήκευση, ανάκτηση δεδομένων, γίνονται χωρίς να χρειαστεί να χτιστεί ολόκληρη υποδομή από την αρχή.

Ένα παράδειγμα: αν χρειάζεται να εγγραφεί κάποιος χρήστης, δεν είναι απαραίτητο να δημιουργηθεί κάποιο API και να γίνει χειροκίνητη POST κλήση. Μπορεί απλά να χρησιμοποιηθεί η μέθοδος supabase.auth.signUp(). Το Supabase χειρίζεται ό,τι χρειάζεται στο παρασκήνιο, δηλαδή ο χρήστης προστίθεται στη βάση, και όλα λειτουργούν χωρίς ιδιαίτερο κόπο. Ακόμα και για επαλήθευση email, υπάρχει έτοιμος τρόπος. Δεν χρειάζονται ρυθμίσεις SMTP ή κάτι άλλο περίπλοκο, απλά δηλώνεται ο πάροχος και η λειτουργία είναι ενεργή.

Το ίδιο απλή είναι και η διαχείριση δεδομένων. Οι βασικές εντολές όπως insert, select ή update χρησιμοποιούνται με τρόπο γνώριμο σαν να πρόκειται για ORM, χωρίς όμως να μπλέκεται κάτι τόσο βαρύ. Τα δεδομένα μπορούν να γραφτούν σε JSON και να περαστούν απευθείας στο UI, χωρίς ενδιάμεσα βήματα.

Αν πρέπει να διαχειριστεί κανείς αρχεία, όπως εικόνες ή PDF, το Supabase προσφέρει έτοιμο αποθηκευτικό χώρο. Δεν χρειάζεται να ενσωματωθεί άλλη υπηρεσία. Η διαδικασία είναι αρκετά απλή ώστε να ενσωματώνεται στην εφαρμογή χωρίς να αποσπά.

Το σημαντικότερο, ίσως, είναι ότι όλα αυτά γίνονται χωρίς πολλές και δύσκολες διαδικασίες. Δεν υπόσχεται τα πάντα, αλλά αυτά που κάνει, τα κάνει ήσυχα και σταθερά. Επιτρέπει στον προγραμματιστή να επικεντρωθεί σε αυτά που έχουν σημασία και αφήνει το υπόλοιπο να δουλεύει στο φόντο. Και κάποιες φορές, αυτό είναι αρκετό.

Η τεκμηρίωση, γενικά, είναι αρκετά φιλική. Υπάρχουν έτοιμα παραδείγματα, και αν υπάρχει ανάλογη εμπειρία χρήσης Firebase ή άλλων παρόμοιων υπηρεσιών, η λογική του Supabase γίνεται σχετικά εύκολα κατανοητή. Για φοιτητικές εφαρμογές ή γρήγορα MVPs, είναι μάλλον ιδανικό, γιατί δεν χάνεται πολύτιμος χρόνος σε ρυθμίσεις, αλλά βοηθάει τους ενδιαφερόμενους να επικεντρωθούν στο πραγματικό περιεχόμενο.

Αυτό που βοηθά επίσης είναι η κονσόλα του Supabase. Από εκεί μπορούν να ελεγχθούν όλες οι εγγραφές των χρηστών που έχουν πραγματοποιήσει εγγραφή στην εφαρμογή, να αλλαχθούν στοιχεία, ή και να διαγραφούν χρήστες, χωρίς να συνταχθεί ξεχωριστό κομμάτι κώδικα. Είναι δηλαδή και developer-friendly αλλά και beginner-friendly, κάτι που σπάνια συνδυάζεται.

3.4.4 Υποστήριξη σε Πραγματικό Χρόνο (Realtime)

Ένα από τα πιο πρακτικά εργαλεία που έχει το Supabase είναι η υποστήριξη για αλλαγές σε πραγματικό χρόνο. Δηλαδή, όταν κάποιος χρήστης προσθέσει ή αλλάξει κάτι στη βάση δεδομένων, μπορεί να εμφανιστεί στην εφαρμογή κατευθείαν, χωρίς να χρειαστεί να γίνει refresh ή να πραγματοποιηθεί έξτρα fetch. Όλο αυτό γίνεται με WebSockets, και πιο συγκεκριμένα, με ένα εργαλείο που λέγεται Realtime [27].

Για όσους έχουν δουλέψει με chat ή λίστες εργασιών, είναι πολύ βολικό να μην χρειάζεται συνεχώς polling. Αντί να γίνεται ερώτηση κάθε 5 δευτερόλεπτα “έγινε κάτι;”, υπάρχει έναν listener που

ενημερώνει αμέσως πότε κάτι αλλάζει. Το μόνο που πρέπει να γίνει είναι να δηλωθεί ενδιαφέρον σε ένα πίνακα και να γίνεται ακρόαση για INSERT, UPDATE ή DELETE.

Στο Flutter, η διαδικασία είναι απλή. Χρησιμοποιείται το .subscribe() και χειρίζεται το stream των αλλαγών. Αν κάποιος χρήστης σβήσει μια εγγραφή, μπορεί να αφαιρεθεί από τη λίστα χωρίς να φορτωθεί ξανά όλη τη βάση. Αυτό μειώνει και την κίνηση στο δίκτυο και κάνει το UI πιο "ζωντανό" και το σημαντικό είναι ότι δεν χρειάζεται να γίνει ρύθμιση με custom socket servers ή να γίνονται ενοχλητικά workarounds. Όλα αυτά δίνονται από το Supabase έτοιμα. Δηλαδή έχουμε real-time χωρίς επιπλέον κόστος, χωρίς τρίτες βιβλιοθήκες και χωρίς ταλαιπωρία με events.

3.4.5 Αποθήκευση και Διαχείριση Αρχείων (Storage)

Μια συχνή ανάγκη σε πολλές εφαρμογές είναι η αποθήκευση αρχείων. Εικόνες, PDF, ήχοι. Συχνά οι χρήστες χρειάζονται να ανεβάσουν τέτοιο υλικό, είτε πρόκειται για προφίλ, είτε για έγγραφα. Το Supabase προσφέρει ένα εργαλείο που καλύπτει αυτή τη λειτουργία, το Supabase Storage, χωρίς να απαιτείται εξωτερική υπηρεσία ή δύσκολη ρύθμιση [27].

Το σύστημα δουλεύει με buckets, που λειτουργούν σαν φάκελοι. Μέσα σε κάθε bucket, τα αρχεία έχουν τη δική τους θέση και μπορούν να οργανωθούν εύκολα. Ο κάθε προγραμματιστής μπορεί να ρυθμίσει ποιος βλέπει τι. Δηλαδή, είναι δυνατό να επιτρέπεται η πρόσβαση μόνο στον χρήστη που ανέβασε ένα αρχείο — και κανέναν άλλον.

Η χρήση του με το Flutter είναι γενικά απλή. Υπάρχει έτοιμη υποστήριξη μέσω της βιβλιοθήκης supabase_flutter, η οποία περιλαμβάνει μεθόδους για αποστολή, λήψη, διαγραφή αρχείων και δημιουργία συνδέσμων προβολής. Ένα κλασικό παράδειγμα είναι η φωτογραφία προφίλ: ο χρήστης την ανεβάζει, το αρχείο αποθηκεύεται, και μετά προβάλλεται μέσω προσωρινού συνδέσμου (signed URL).

Ένα πρακτικό χαρακτηριστικό είναι η δυνατότητα να οριστεί πόσο θα διαρκεί αυτός ο σύνδεσμος. Έτσι, αντί το αρχείο να είναι προσβάσιμο για πάντα, η πρόσβαση λήγει, κάτι χρήσιμο όταν πρόκειται για ευαίσθητα δεδομένα. Ο σύνδεσμος μπορεί να λειτουργεί για λίγα λεπτά ή λίγες ώρες, ανάλογα με τις ανάγκες.

Αν και δεν έχει όλες τις δυνατότητες πιο ώριμων λύσεων, όπως το Firebase Storage, το Supabase Storage είναι επαρκές για τις πιο συνηθισμένες περιπτώσεις. Το μεγάλο του πλεονέκτημα είναι ότι συνεργάζεται άψογα με το υπόλοιπο οικοσύστημα του Supabase. Δεν χρειάζεται να δημιουργούνται ξεχωριστοί μηχανισμοί ασφαλείας· μπορεί να συνδεθεί απευθείας με χρήστες και δεδομένα της βάσης.

Για περιπτώσεις όπως φοιτητικές εφαρμογές, όπου υπάρχει ανάγκη αποστολής αρχείων (φωτογραφιών, εγγράφων, εργασιών, βιογραφικών), η λύση αυτή είναι ιδανική. Δίνει τη δυνατότητα να υλοποιηθεί ένα backend πλήρες, χωρίς να χρησιμοποιηθούν τρίτες υπηρεσίες ή να χρησιμοποιηθούν περίπλοκα API.

Κεφάλαιο 4ο: Ανάπτυξη διαδραστικής εφαρμογής για μικρό τεχνολογικό μουσείο

4.1 Εισαγωγή

Σε αυτή την ενότητα παρουσιάζεται μια εφαρμογή που σχεδιάστηκε για να βοηθήσει τους επισκέπτες του μικρού τεχνολογικού μουσείου του ΔΙΠΑΕ, να συνδεθούν πιο εύκολα με τα εκθέματα. Ο βασικός σκοπός ήταν να γίνει η εμπειρία πιο απλή, χωρίς να χρειάζεται ο επισκέπτης να έχει γνώσεις από τεχνολογία ή να ακολουθεί περίπλοκες οδηγίες.

Η χρήση της εφαρμογής είναι πολύ εύκολη. Κάθε έκθεμα έχει δίπλα του έναν QR κωδικό. Ο επισκέπτης τον σαρώνει με το κινητό του και αμέσως βλέπει στην οθόνη πληροφορίες που σχετίζονται με αυτό που βλέπει. Έτσι δεν χρειάζεται να ρωτήσει κάποιον ή να ψάξει κάτι μόνος του.

Εκτός από τις πληροφορίες, υπάρχουν και σύντομα κουίζ μέσα στην εφαρμογή. Αυτά υλοποιήθηκαν για να τραβήξουν το ενδιαφέρον και να κάνουν τον επισκέπτη να σκεφτεί λίγο παραπάνω, χωρίς να είναι κουραστικά ή απαιτητικά.

Η εμφάνιση της εφαρμογής είναι απλή. Το μενού είναι καθαρό, οι επιλογές λίγες και κατανοητές, και όλα έχουν σχεδιαστεί με στόχο να μη δυσκολεύεται ο χρήστης. Δεν προσπαθεί να κάνει εντύπωση με τεχνικά στοιχεία, αλλά λειτουργεί απλά και κάνει τη δουλειά της.

Συνολικά, η εφαρμογή είναι εκεί όταν τη χρειάζεται ο επισκέπτης. Δεν ενοχλεί, δεν τον πιέζει και δεν μπαίνει ανάμεσα σε αυτόν και την εμπειρία της επίσκεψης στο μουσείο. Λειτουργεί περισσότερο σαν ένας βοηθός που είναι «αόρατος» και εμφανίζεται μόνο όταν χρειάζεται.

Τεχνικά λοιπόν, για να δημιουργηθεί η εφαρμογή χρησιμοποιήθηκε το Flutter για το μπροστινό μέρος (αυτό που βλέπει ο χρήστης) και το Supabase για τα δεδομένα και τις λειτουργίες στο παρασκήνιο. Επιλέχθηκαν αυτά τα εργαλεία επειδή είναι απλά στη χρήση και δεν χρειάζονται εξειδικευμένες γνώσεις για να στηθούν. Συνεργάζονται καλά μεταξύ τους, κι αυτό βοηθά ώστε όλα να λειτουργούν χωρίς προβλήματα και να συνδέονται σωστά με τη βάση δεδομένων.

Μια βασική επιλογή στον σχεδιασμό ήταν να αποφευχθεί η φόρτωση της αρχικής οθόνης με πολλές πληροφορίες. Ο χρήστης βλέπει μόνο τα απολύτως απαραίτητα, όπως πού να πατήσει για πληροφορίες, κουίζ ή αλλαγή γλώσσας με αποτέλεσμα να μην μπερδεύεται. Η πρώτη φορά που κάποιος ανοίγει την εφαρμογή είναι απλή και ξεκάθαρη, δεν χρειάζεται να ψάχνει.

Το κοινό στο οποίο απευθύνεται η εφαρμογή είναι ευρύ. Δεν έχει σημασία αν ο χρήστης είναι μικρός ή μεγάλος, ή αν έχει επαφή με την τεχνολογία. Όλοι μπορούν να την καταλάβουν. Η γραμματοσειρά είναι καθαρή, οι επιλογές είναι λίγες και λογικές, και γενικά το περιβάλλον είναι άμεσο.

Ένα ακόμα βασικό σημείο είναι η αλλαγή γλώσσας. Ανάλογα με το τι γλώσσα έχει το κινητό του χρήστη, η εφαρμογή εμφανίζεται στα ελληνικά ή στα αγγλικά. Προς το παρόν υπάρχουν μόνο αυτές οι δύο, αλλά μπορεί εύκολα να προστεθούν κι άλλες αν χρειαστεί.

Επίσης, έχει προβλεφθεί να είναι εύκολο να γίνουν αλλαγές ή προσθήκες στο μέλλον. Αν προστεθούν νέα εκθέματα ή νέες λειτουργίες, δεν χρειάζεται να ξαναγραφτεί όλη η εφαρμογή από την αρχή. Η βάση δεδομένων είναι φτιαγμένη έτσι ώστε να μπορείς να προσθέσεις υλικό με απλές αλλαγές.

Στο επόμενο μέρος, θα αναλυθούν τα βασικά κομμάτια της εφαρμογής: τι βλέπει ο χρήστης όταν την ανοίγει, πώς σαρώνει QR κωδικούς, πώς εμφανίζονται οι πληροφορίες, πώς λειτουργούν τα κουίζ,

πώς γίνεται η εναλλαγή γλώσσας και ποια είναι η γενική αισθητική. Θα γίνει επίσης αναφορά στο πώς έχει στηθεί η βάση δεδομένων και πώς επικοινωνεί με την εφαρμογή σε πραγματικό χρόνο.

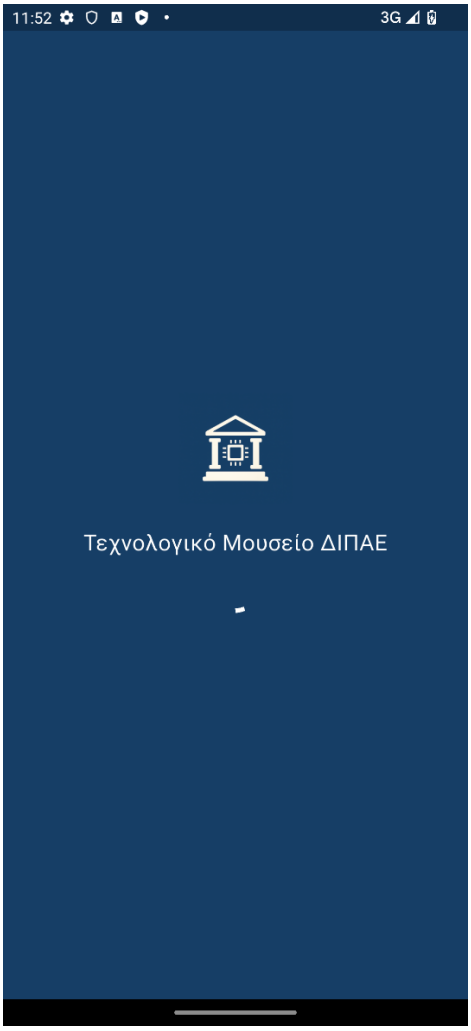
Ο στόχος είναι να φανεί καθαρά πώς δουλεύει το σύστημα, τόσο για τον απλό χρήστη όσο και για όποιον θέλει να το βελτιώσει ή να το χρησιμοποιήσει αλλού.

4.2 Περιγραφή λειτουργιών εφαρμογής

Σε αυτήν την ενότητα θα περιγραφούν λεπτομερώς όλες οι λειτουργίες της εφαρμογής καθώς και το πως κάθε μια από αυτές συμβάλλει στην καλύτερη δυνατή εμπειρία για τους επισκέπτες του μουσείου.

4.2.1 Αρχική Οθόνη και Πλοήγηση Χρήστη

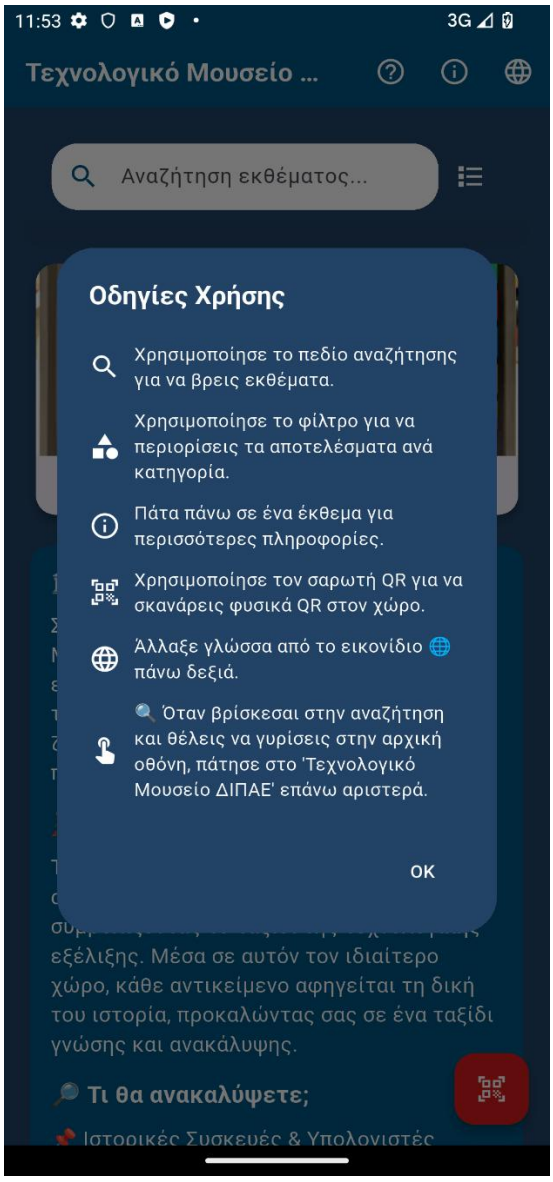
Η πρώτη σελίδα μιας εφαρμογής είναι το σημείο από όπου ξεκινάνε όλα. Είναι το πρώτο πράγμα που βλέπει ο χρήστης και, συχνά, από αυτήν σχηματίζει άποψη για το πόσο εύκολη ή δύσκολη θα είναι η υπόλοιπη εμπειρία. Αν κάτι είναι απλό από την αρχή, υπάρχει μεγάλη πιθανότητα να συνεχίσει να χρησιμοποιείται χωρίς προβλήματα. Αν, όμως, μπερδεύει ή καθυστερεί, μπορεί να αποθαρρύνει τον επισκέπτη από την πρώτη κιόλας στιγμή.



Εικόνα 4.1: Οθόνη εκκίνησης της εφαρμογής

Στην παραπάνω εικόνα φαίνεται το τι εμφανίζεται στον χρήστη μόλις εκκινήσει την εφαρμογή από το κινητό του. Σε περίπτωση που χρησιμοποιεί οποιαδήποτε άλλη γλώσσα, πέρα από την ελληνική, στο κινητό του, τότε θα εμφανιστεί η ίδια ακριβώς οθόνη αλλά αντί για ‘Τεχνολογικό Μουσείο ΔΙΠΑΕ’ θα γράφει ‘IHU Tech Museum’. Σε περιβάλλοντα όπως τα μουσεία, ο χρόνος των επισκεπτών είναι περιορισμένος. Οι περισσότεροι θέλουν να βλέπουν τα εκθέματα και όχι να ασχολούνται με οθόνες. Γι’ αυτό, η αρχική σελίδα της εφαρμογής πρέπει να είναι όσο πιο καθαρή και εύκολη γίνεται. Δεν υπάρχει περιθώριο για δοκιμές ή για πολύπλοκα μενού που οδηγούν σε άλλες σελίδες χωρίς σαφή προορισμό.

Η διάταξη που ακολουθήθηκε είναι κάθετη. Τα βασικά κουμπιά είναι τοποθετημένα το ένα κάτω από το άλλο, με αρκετό κενό ανάμεσά τους. Έτσι, αποφεύγονται λάθη στην αφή, ειδικά από χρήστες με λιγότερη εμπειρία ή που χειρίζονται μικρότερες οθόνες. Όλα είναι στη θέση τους, χωρίς περιττές μετακινήσεις ή κρυμμένες λειτουργίες. Η πλοήγηση γίνεται με έναν φυσικό και ήρεμο ρυθμό. Δεν υπάρχουν «εκπλήξεις» ή νέα παράθυρα που εμφανίζονται ξαφνικά και μπερδεύουν τον χρήστη.

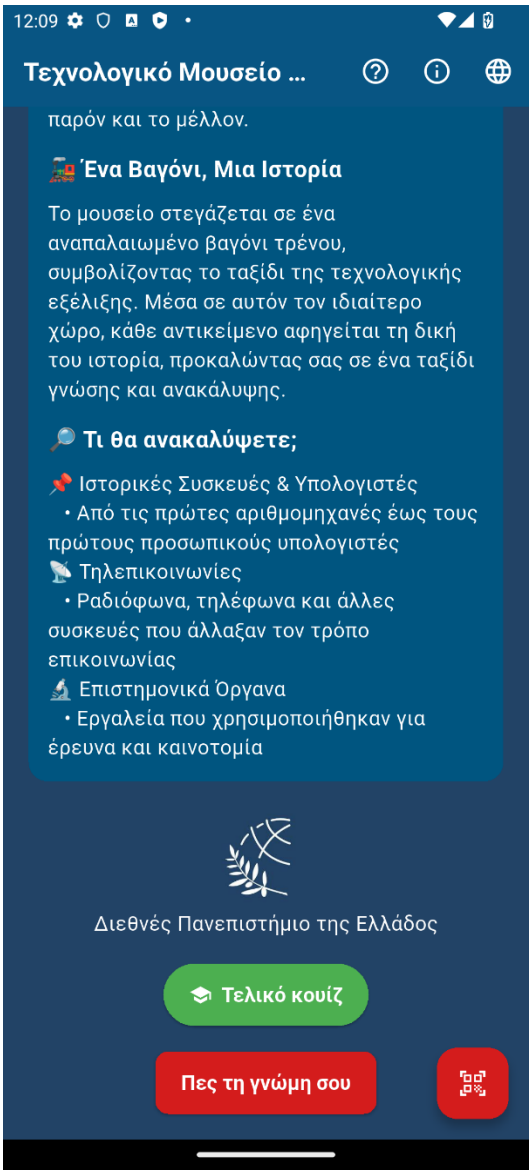


Εικόνα 4.2: Πρώτη εκκίνηση εφαρμογής. Διακρίνεται παράθυρο με οδηγίες σωστής χρήσης

Στην εικόνα 4.2 βλέπουμε την πρώτη εκκίνηση της εφαρμογής μετά την εγκατάσταση. Εμφανίζεται αυτόματα το παράθυρο οδηγιών για τον χρήστη, δείχνοντας του πως μπορεί να χρησιμοποιήσει την εφαρμογή. Φυσικά αυτό το παράθυρο δεν εμφανίζεται κάθε φορά που ο χρήστης θα εκκινήσει την εφαρμογή, αλλά μπορεί να ανατρέξει πίσω σε αυτό με το κουμπί του αγγλικού ερωτηματικού που βρίσκεται επάνω δεξιά μαζί με τα υπόλοιπα κουμπιά. Το περιβάλλον είναι φωτεινό, όχι μόνο για λόγους αισθητικής, αλλά και πρακτικούς. Σε έναν χώρο όπως ένα μουσείο, το φως αλλάζει συνεχώς, και οθόνες με σκούρο φόντο ή περίπλοκα χρώματα μπορεί να μην είναι ευανάγνωστες. Εδώ, τα γράμματα είναι μεγάλα και καθαρά, το φόντο δεν μπερδεύει και τα χρώματα των κουμπιών είναι ήπια, αλλά ξεκάθαρα. Όλα έχουν δημιουργηθεί για να λειτουργούν σωστά.

Οι βασικές επιλογές που προσφέρει η αρχική σελίδα είναι τρεις: σάρωση QR, πρόσβαση σε κουίζ και εμφάνιση πληροφοριών. Δεν υπάρχουν ενδιάμεσες οθόνες ή δευτερεύοντα μενού. Κάθε λειτουργία οδηγεί κατευθείαν σε αυτό που προσφέρει. Έτσι, ο χρήστης δεν χάνει χρόνο και ξέρει πάντα πού βρίσκεται.

Εικόνα 4.3: Αρχική οθόνη εφαρμογής



Εικόνα 4.4: Αρχική οθόνη εφαρμογής (συνέχεια)

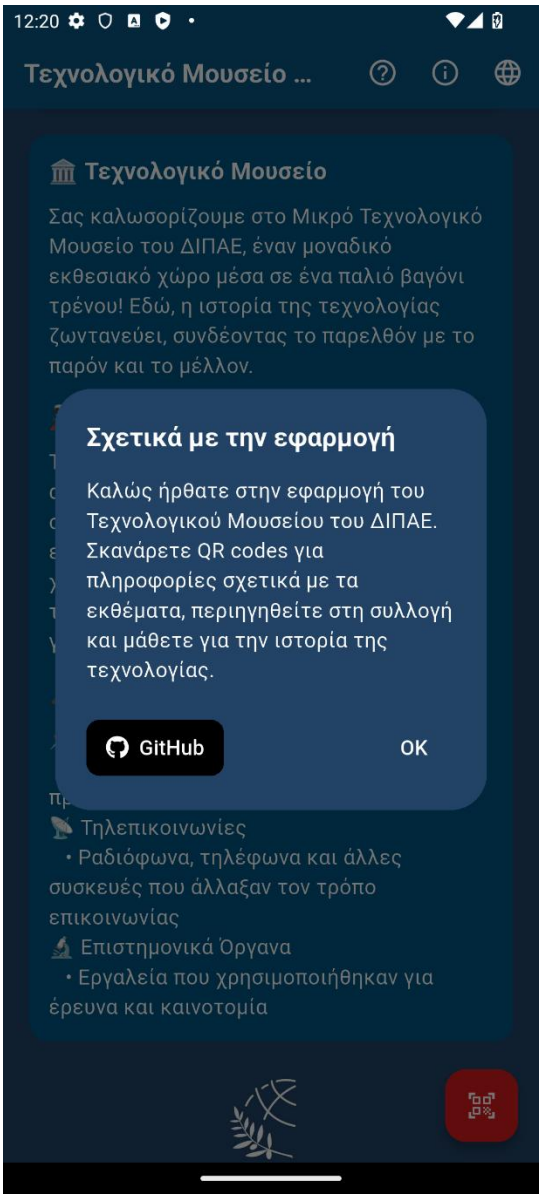
Στις εικόνες 4.3 και 4.4 διακρίνεται ολόκληρη η αρχική οθόνη της εφαρμογής. Διακρίνονται ο τίτλος της εφαρμογής επάνω αριστερά, τα κουμπιά οδηγιών, πληροφοριών σχετικές με την εφαρμογή και αλλαγής γλώσσας καθώς και η μπάρα αναζήτησης με την επιλογή κατηγορίας δεξιά της, το τυχαίο έκθεμα του μουσείου που προτείνει η εφαρμογή, λίγα λόγια για το μουσείο, το λογότυπο του Διεθνούς Πανεπιστημίου της Ελλάδας και τα κουμπιά του τελικού κουίζ, αξιολόγησης της εφαρμογής και το κουμπί για σάρωση QR κωδικών των εκθεμάτων. Αν θελήσει ο χρήστης να γυρίσει πίσω, έχει επιλογές. Μπορεί να χρησιμοποιήσει το κουμπί «πίσω» της συσκευής του ή να πατήσει το εμφανές κουμπί που υπάρχει σε κάθε σελίδα. Με αυτόν τον τρόπο, δεν μένει ποτέ κολλημένος κάπου. Νιώθει ότι έχει τον έλεγχο και δεν χρειάζεται να μάθει πώς δουλεύει η εφαρμογή γιατί ήδη λειτουργεί με τρόπο που είναι ήδη οικείος.

Από τεχνικής άποψης, η εφαρμογή ξεκινά από ένα βασικό αρχείο όπου ορίζεται ποια σελίδα είναι η πρώτη που θα δει ο χρήστης. Από εκεί, έχουν καθοριστεί διαδρομές που επιτρέπουν τη μετακίνηση ανάμεσα στις σελίδες. Κάθε λειτουργία έχει το δικό της μονοπάτι και αυτό διευκολύνει την προσθήκη

νέων χαρακτηριστικών στο μέλλον. Αν χρειαστεί να προστεθεί μια νέα σελίδα, γίνεται εύκολα χωρίς να χαλάσει το υπάρχον σύστημα.

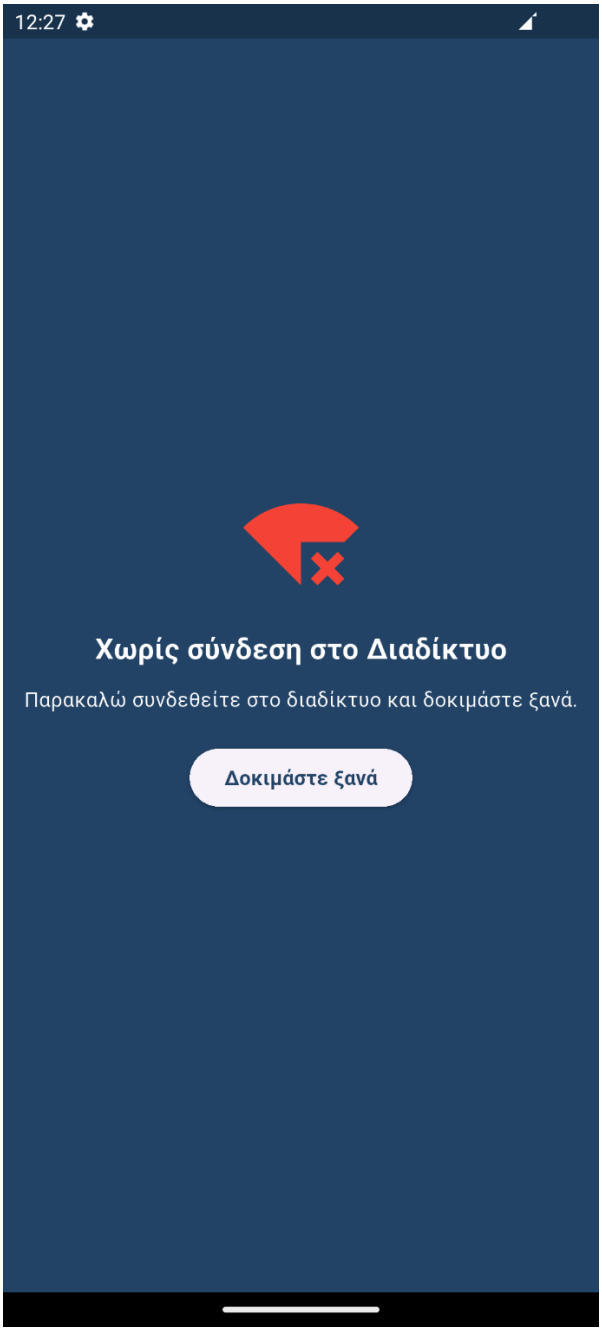
Η εφαρμογή ανιχνεύει αυτόματα τη γλώσσα του κινητού. Αν το τηλέφωνο είναι στα ελληνικά, η εφαρμογή εμφανίζεται στα ελληνικά. Αν είναι στα αγγλικά, τότε το περιεχόμενο αλλάζει ανάλογα. Αυτή η λειτουργία βοηθά και τους επισκέπτες που δεν μιλούν ελληνικά. Αν χρειαστεί να μπουν κι άλλες γλώσσες, αυτό μπορεί να γίνει χωρίς να χρειαστεί να αλλάξει η δομή ολόκληρης της εφαρμογής.

Έχει προβλεφθεί και η υποστήριξη για άτομα που δυσκολεύονται με την όραση ή τη χρήση συσκευών αφής. Τα κουμπιά έχουν μέγεθος που πατιέται εύκολα, και υπάρχει αρκετός χώρος μεταξύ τους για να μη γίνονται λάθη. Τα χρώματα είναι επιλεγμένα ώστε να καλύπτουν και περιπτώσεις χρωματικής αδυναμίας. Αν και δεν έχουν προστεθεί ακόμα λειτουργίες όπως φωνητική ανάγνωση ή χρήση χειρονομιών, η εφαρμογή είναι έτοιμη να τις δεχτεί στο μέλλον, χωρίς να χρειαστεί ριζικές αλλαγές.



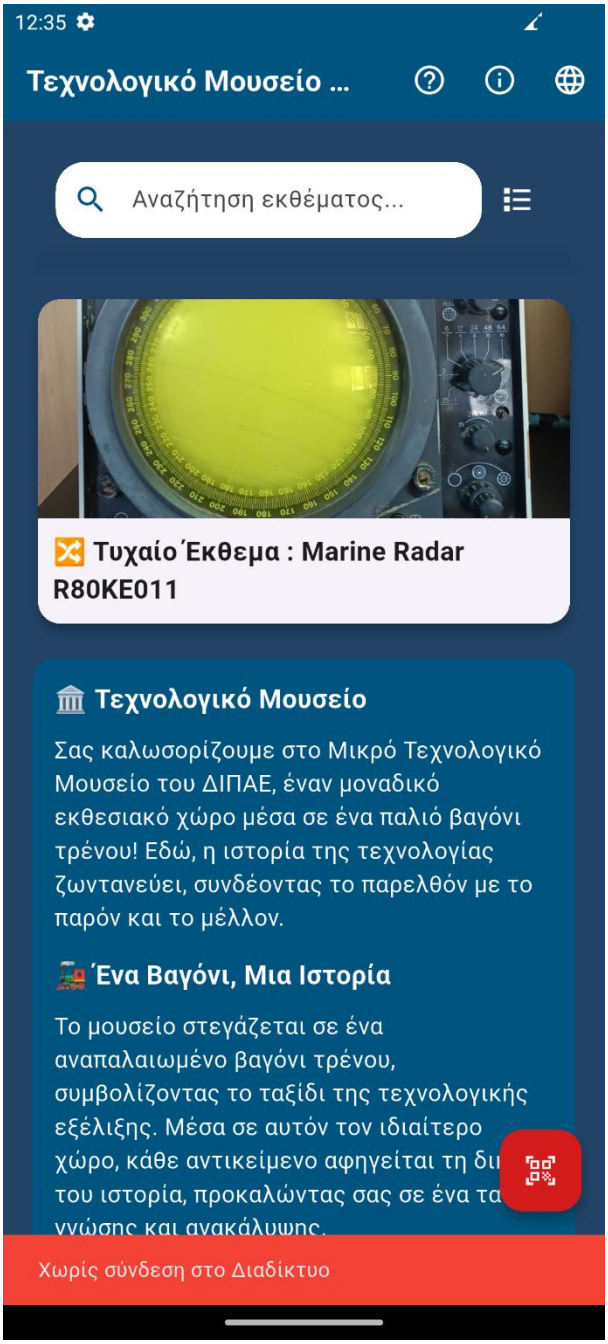
Εικόνα 4.5: Παράθυρο πληροφοριών σχετικές με την εφαρμογή

Παραπάνω φαίνεται το παράθυρο πληροφοριών σχετικά με την εφαρμογή. Αυτό εμφανίζεται όποτε ο χρήστης πατήσει το κουμπί info. Εξηγεί τι μπορεί να κάνει ο χρήστης με την εφαρμογή καθώς και να δει τον κώδικα της εφαρμογής που βρίσκεται στο GitHub. Η αρχική σελίδα μπορεί να θεωρηθεί ως σημείο αναφοράς. Αν ο χρήστης χαθεί ή μπερδευτεί, μπορεί πάντα να επιστρέψει εκεί. Αυτό δίνει σιγουριά και κάνει την πλοήγηση πιο ξεκούραστη. Σε έναν χώρο όπως το μουσείο, όπου ο επισκέπτης έχει πολλούς περισπασμούς και λίγο χρόνο, κάτι τέτοιο κάνει μεγάλη διαφορά.



Εικόνα 4.6: Εκκίνηση της εφαρμογής χωρίς σύνδεση στο διαδίκτυο

Στην παραπάνω εικόνα φαίνεται η προσπάθεια εκκίνησης της εφαρμογής με απουσία διαδικτύου (WiFi ή δεδομένα κινητής τηλεφωνίας). Όλα τα δεδομένα της εφαρμογής φορτώνονται από τη βάση δεδομένων, οπότε αν γινόταν εκκίνηση χωρίς έλεγχο ύπαρξης σύνδεσης στο διαδίκτυο, απλά δε θα εμφανιζόταν τίποτα σαν αρχική οθόνη. Επίσης, δίνεται η δυνατότητα επανάληψης εκκίνησης, έπειτα από επιτυχή επανασύνδεση στο διαδίκτυο, χωρίς να χρειάζεται έξοδος και επανεκκίνηση της εφαρμογής.



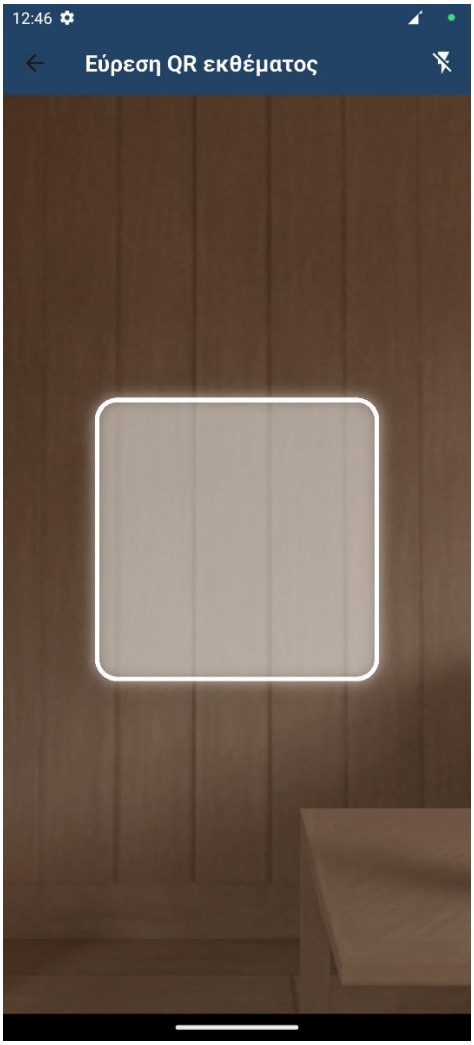
Εικόνα 4.7: Απουσία σύνδεσης στο διαδίκτυο

Η σύνδεση στο διαδίκτυο μπορεί να διακοπεί ακόμα και κατά τη διάρκεια χρήσης της εφαρμογής (αφού έχει φορτωθεί η αρχική σελίδα). Ο χρήστης μπορεί να ενημερωθεί για αυτό με διακριτικό μήνυμα στο κάτω μέρος της οθόνης, όπως φαίνεται στο παράδειγμα της εικόνας 4.7.

Τέλος, ολόκληρη η αρχική οθόνη έχει σχεδιαστεί με προσοχή ώστε να μπορεί να επεκταθεί. Αν προστεθούν νέες λειτουργίες ή εμφανιστούν διαφορετικές ανάγκες, μπορεί να προσαρμοστεί εύκολα χωρίς να χαθεί η σταθερότητα που έχει ήδη επιτευχθεί. Από την πρώτη χρήση, ο επισκέπτης μπορεί να κατανοήσει την χρήση της εφαρμογής και να αντιληφθεί τις πολλαπλές λειτουργίες της χωρίς να του το εξηγήσει κανείς.

4.2.2 Σάρωση QR και Πληροφορίες Εκθεμάτων

Όταν κάποιος επισκέπτεται ένα μουσείο, τις περισσότερες φορές κοιτάζει τα εκθέματα και θέλει να μάθει κάτι παραπάνω. Στα περισσότερα μουσεία, βλέπει κανείς μικρές καρτέλες με βασικά στοιχεία, αλλά συνήθως το περιεχόμενό τους είναι περιορισμένο. Άλλοτε υπάρχουν και ξεναγοί ή φυλλάδια. Όμως αυτά, πρακτικά, δεν βολεύουν κάθε επισκέπτη. Κάποιος μπορεί να έχει δυσκολία στην όραση, να μη θέλει να κουβαλάει έντυπο υλικό ή απλώς να μην προλαβαίνει να διαβάσει επί τόπου. Σε τέτοιες περιπτώσεις, οι QR κωδικοί μπορούν να δώσουν μια πρακτική λύση.



Εικόνα 4.8: Οθόνη σάρωσης QR με χρήση κάμερας

Η τεχνολογία δεν είναι καινούργια. Οι QR χρησιμοποιούνται εδώ και αρκετά χρόνια, αλλά σήμερα οι περισσότεροι έχουν εξοικειωθεί μαζί τους. Ανοίγεις την κάμερα στο κινητό, εστιάζεις σε έναν μικρό τετράγωνο κώδικα και σχεδόν αμέσως εμφανίζεται σχετική πληροφορία στην οθόνη. Στην περίπτωση της εφαρμογής που δημιουργήθηκε για το μουσείο, η χρήση QR εφαρμόστηκε ώστε οι επισκέπτες να μπορούν να μαθαίνουν περισσότερα χωρίς να μπαίνουν σε διαδικασία πληκτρολόγησης ή αναζήτησης.

Ήδη από την αρχική οθόνη, ο χρήστης μπορεί να δει την επιλογή “Σάρωση Κωδικού”. Δεν χρειάζεται να την ψάξει μέσα σε μενού. Είναι σε εμφανές σημείο. Με το πάτημά της, ανοίγει η κάμερα και προβάλλεται στο κέντρο της οθόνης ένα πλαίσιο που δείχνει πού να τοποθετηθεί ο κώδικας. Το μόνο που έχει να κάνει ο επισκέπτης είναι να φέρει τον QR εντός του πλαισίου. Δεν χρειάζεται να πατήσει “επόμενο” ή “συνέχεια”.

Οι QR έχουν τοποθετηθεί με τρόπο που δεν επηρεάζει την αισθητική των εκθεμάτων. Είναι αρκετά μικροί, αλλά ταυτόχρονα εύκολα αναγνωρίσιμοι. Συνήθως είναι χαμηλά, κολλημένοι σε διακριτικά σημεία, ώστε να μην τραβούν άσκοπα την προσοχή. Δεν χρειάζονται επιπλέον εξαρτήματα, οθόνες ή εξειδικευμένος εξοπλισμός. Το μόνο απαραίτητο είναι το κινητό που ούτως ή άλλως έχει ο επισκέπτης μαζί του.

Από τη στιγμή που η εφαρμογή αναγνωρίσει τον QR, εντοπίζει αυτόματα το έκθεμα με το οποίο συνδέεται και μεταφέρει τον χρήστη στην κατάλληλη σελίδα. Για παράδειγμα, αν κάποιος σαρώνει έναν κωδικό που αντιστοιχεί σε ένα παλιό τεχνολογικό αντικείμενο, θα μεταφερθεί στη σελίδα με το σχετικό κείμενο, τις εικόνες, τις ημερομηνίες και κάθε άλλο διαθέσιμο υλικό. Ο χρήστης μεταφέρεται σχεδόν αμέσως εκεί που χρειάζεται. Δεν χρειάζεται να περιμένει ούτε να επιβεβαιώσει τίποτα.

Το σύστημα αυτό λειτουργεί με τη βοήθεια μιας μικρής βιβλιοθήκης μέσα στην εφαρμογή, η οποία ενεργοποιεί την κάμερα και διαβάζει τον κώδικα. Είναι ελαφριά, σταθερή και δεν χρειάζεται επιπλέον εγκαταστάσεις. Μόλις εντοπιστεί ο QR, η εφαρμογή αναγνωρίζει τον σύνδεσμο ή τον κωδικό και δείχνει στην οθόνη το σχετικό περιεχόμενο. Ο χρήστης δεν χρειάζεται να κάνει κάτι άλλο.

Ο βασικός λόγος που χρησιμοποιήθηκε αυτή η τεχνολογία ήταν επειδή τα πράγματα έπρεπε να μείνουν απλά. Δεν ήταν σκοπός να εντυπωσιάσει, αλλά να διευκολύνει. Δεν υπάρχει ανάγκη για επιπλέον εξοπλισμό. Ούτε πολύπλοκα μενού. Η προσέγγιση βασίστηκε στην ιδέα “λιγότερα βήματα, λιγότερα προβλήματα”.

Υπήρχαν και κάποιες τεχνικές παρατηρήσεις. Σε ορισμένα σημεία του μουσείου όπου ο φωτισμός δεν ήταν επαρκής, χρειαζόταν λίγη περισσότερη προσπάθεια για να εντοπιστεί ο QR. Αυτό όμως αντιμετωπίστηκε εύκολα, αφού η εφαρμογή μπορεί να ενεργοποιήσει τον φακό της συσκευής, αν χρειαστεί, μόνο στην έκδοση android, καθώς δεν υπάρχει δυνατότητα ελέγχου του hardware μέσω web.

Γενικά, η λειτουργία αυτή έγινε αποδεκτή θετικά. Ο επισκέπτης μπορεί να τη χρησιμοποιήσει ή να την αγνοήσει. Δεν του επιβάλλεται. Δεν τον περιορίζει. Αν θέλει, απλώς σαρώνει και διαβάζει. Αν όχι, συνεχίζει την επίσκεψη κανονικά. Η τεχνολογία δεν παρεμβάλλεται απλά υπάρχει στο παρασκήνιο και εμφανίζεται μόνο όταν χρειαστεί. Αυτό ήταν το ζητούμενο από την αρχή.

Μόλις ολοκληρωθεί η σάρωση ενός QR κωδικού, η εφαρμογή προβάλλει μια νέα σελίδα με υλικό σχετικό με το αντικείμενο που αντιστοιχεί στον κωδικό. Δεν χρειάζεται καμία επιπλέον ενέργεια από τον χρήστη. Η μετάβαση γίνεται αυτόματα. Το περιεχόμενο εμφανίζεται άμεσα και είναι διαμορφωμένο έτσι ώστε να διαβάζεται άνετα σε κινητές συσκευές.

Εικόνα 4.9: Αποτέλεσμα σάρωσης QR

Στην περίπτωση όπου το QR που σαρώνει ο χρήστης, δεν αντιστοιχεί σε έκθεμα του μουσείου, τότε ενημερώνεται με κατάλληλο μήνυμα και παραμένει στην οθόνη σάρωσης. Πάνω στην οθόνη εμφανίζεται το όνομα ή ο τίτλος του εκθέματος. Ακολουθεί ένα σύντομο κείμενο που περιγράφει βασικά στοιχεία, όπως τι είναι το έκθεμα, ποια είναι η χρήση του, από ποια εποχή προέρχεται και ποιος είναι ο λόγος που βρίσκεται στο μουσείο. Το ύφος του κειμένου είναι απλό, χωρίς τεχνική ορολογία που θα δυσκόλευε την κατανόηση. Έτσι, γίνεται προσβάσιμο σε επισκέπτες κάθε ηλικίας.

Σε δεύτερη θέση εμφανίζονται εικόνες. Οι περισσότερες δείχνουν το αντικείμενο όπως είναι σήμερα, αλλά μπορεί να υπάρχουν και φωτογραφίες από παλιότερες εποχές ή παραδείγματα χρήσης του. Οι εικόνες τοποθετούνται με τέτοιο τρόπο ώστε να μην επιβαρύνουν την προβολή, ούτε να καλύπτουν το κείμενο. Δεν απαιτείται κύλιση για να τις δει κανείς· εμφανίζονται φυσικά μέσα στη ροή της πληροφορίας.

Κάτω από τις βασικές πληροφορίες, υπάρχουν προαιρετικές ενότητες. Ο επισκέπτης μπορεί να πατήσει ένα κουμπί που γράφει “Πίσω” για να μετακινηθεί στην αρχική οθόνη. Επίσης, εμφανίζεται και η δυνατότητα συμμετοχής σε μικρό κουίζ. Οι ερωτήσεις βασίζονται στις πληροφορίες της σελίδας και λειτουργούν περισσότερο σαν παιχνίδι ή μικρή πρόκληση για όποιον επιθυμεί να δοκιμάσει τι θυμάται. Αυτή η επιλογή είναι διαθέσιμη σε όλα τα εκθέματα, αλλά δεν είναι υποχρεωτική να γίνει από τον χρήστη. Απλά είναι εκεί για όποιον θέλει να την δοκιμάσει.

Το περιεχόμενο δεν είναι μόνιμα αποθηκευμένο στο κινητό. Αντίθετα, κάθε φορά που γίνεται σάρωση, η εφαρμογή αντλεί τις πληροφορίες από έναν απομακρυσμένο διακομιστή. Αυτό σημαίνει ότι οι υπεύθυνοι του μουσείου μπορούν να αλλάζουν το περιεχόμενο, να προσθέτουν καινούργια στοιχεία ή να διορθώνουν λάθη χωρίς να χρειαστεί ο επισκέπτης να κάνει αναβάθμιση της εφαρμογής. Η αλλαγή γίνεται στη βάση και εμφανίζεται σε κάθε κινητό που χρησιμοποιεί τη λειτουργία.

Για να μετακινηθεί ο επισκέπτης πίσω στην αρχική οθόνη, αρκεί να πατήσει το πλήκτρο επιστροφής της συσκευής ή το αντίστοιχο κουμπί στο πάνω μέρος της εφαρμογής. Οι διαδρομές μέσα στο περιβάλλον είναι ξεκάθαρες και δεν υπάρχουν μπερδεμένα μενού. Όλα έχουν σχεδιαστεί έτσι ώστε να είναι όσο πιο φιλικά γίνεται, ακόμη και για εκείνους που δεν έχουν μεγάλη εμπειρία με εφαρμογές.

Η εφαρμογή μπορεί να εμφανίσει το περιεχόμενο σε διαφορετικές γλώσσες, ανάλογα με τις ρυθμίσεις του κινητού. Αν η συσκευή είναι ρυθμισμένη στα αγγλικά ή σε οποιαδήποτε άλλη γλώσσα εκτός της ελληνικής, το περιεχόμενο εμφανίζεται στα αγγλικά. Αν είναι στα ελληνικά, τότε εμφανίζεται το ίδιο κείμενο μεταφρασμένο. Η αλλαγή γίνεται αυτόματα. Δεν χρειάζεται να πατήσει κάποιος κάποιο κουμπί ή να ψάξει στις ρυθμίσεις.

Στην ουσία, ο σκοπός της σελίδας που προβάλλεται μετά τη σάρωση είναι να υποστηρίξει τον επισκέπτη στην κατανόηση του εκθέματος. Δεν προσπαθεί να τον εντυπωσιάσει με εντυπωσιακά γραφικά ή περίπλοκα εφέ. Αντίθετα, επιδιώκει να κρατήσει το ενδιαφέρον του με ήρεμο και κατανοητό τρόπο. Η πληροφορία παρουσιάζεται χωρίς πίεση και χωρίς περιττές παρεμβολές.

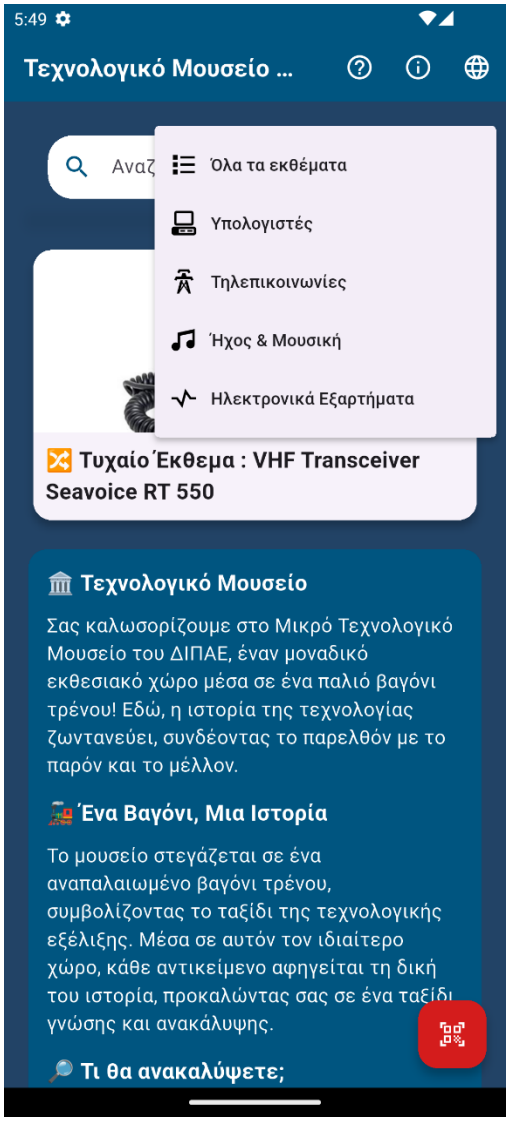
Ο επισκέπτης δεν είναι υποχρεωμένος να ακολουθήσει κάποιον συγκεκριμένο δρόμο. Αν θέλει, μπορεί να δει μόνο τα βασικά. Αν πάλι τον ενδιαφέρει περισσότερο το περιεχόμενο, έχει τη δυνατότητα να προχωρήσει σε βάθος. Η εφαρμογή δεν κατευθύνει ούτε περιορίζει. Απλώς δίνει επιλογές και αφήνει τον καθένα να αποφασίσει πώς θα κινηθεί. Αυτή η ευελιξία ίσως να είναι και ο λόγος που την κάνει προσιτή σε διαφορετικά είδη κοινού, χωρίς να κουράζει ή να επιβάλλεται.

Το μουσείο, εξάλλου, είναι κατεξοχήν χώρος εξερεύνησης. Δεν χρειάζεται οθόνες παντού ή διαρκείς οδηγίες. Ο επισκέπτης βλέπει, σκέφτεται, μπορεί να αναρωτηθεί και να θελήσει να μάθει περισσότερα — αλλά όχι απαραίτητα με τον ίδιο τρόπο κάθε φορά. Εκεί βρίσκει εφαρμογή η ιδέα ενός QR κωδικού δίπλα σε ένα έκθεμα. Είναι διακριτικός. Δεν τραβά την προσοχή και δεν ενοχλεί. Είναι εκεί για όποιον θέλει να τον χρησιμοποιήσει.

Αν ο επισκέπτης δεν χρησιμοποιήσει την εφαρμογή, δεν αλλάζει κάτι ουσιαστικό στην εμπειρία του. Η εφαρμογή δεν διεκδικεί κεντρικό ρόλο. Υπάρχει απλώς ως υποστήριξη, διακριτικά στο παρασκήνιο, έτοιμη να προσφέρει βοήθεια μόνο αν και όταν χρειαστεί. Δεν προϋποθέτει κάποια προετοιμασία, ούτε απαίτηση να έχει ανοιχτεί από πριν. Το μόνο που χρειάζεται είναι μια κάμερα για μια σάρωση που αρκεί για να έχει κανείς πρόσβαση στο υλικό που τον ενδιαφέρει.

Αυτός ο τρόπος λειτουργίας ταιριάζει σε πολλούς τύπους επισκεπτών. Κάποιος που έχει διάθεση και χρόνο μπορεί να διαβάσει περισσότερα. Κάποιος άλλος που απλώς περνάει γρήγορα μπορεί να ρίξει μια ματιά και να προχωρήσει. Δεν υπάρχει σωστό ή λάθος. Ο καθένας ζει την εμπειρία με τον δικό του ρυθμό. Σε σχολεία, η ίδια δυνατότητα μπορεί να λειτουργήσει σαν κάτι έξτρα. Όχι σαν υποχρέωση. Δεν αντικαθιστά τη δασκάλα ή το βιβλίο, αλλά ίσως δίνει έναυσμα. Πιο πολύ για να δοκιμάσει το παιδί κάτι μόνο του. Να νιώσει ότι ανακαλύπτει. Όχι ότι πρέπει να ακολουθήσει ένα σχέδιο.

Σε μια ημέρα με πολλούς επισκέπτες, αυτός ο τρόπος λειτουργίας μειώνει την πίεση στον χώρο. Δεν μαζεύονται όλοι μπροστά από ένα ταμπλό ή μια οθόνη. Ο καθένας με το κινητό του μπορεί να διαβάσει, να δει, να συμμετέχει ή να αγνοήσει.



Εικόνα 4.10: Εμφάνιση κατηγοριών εκθεμάτων

Όταν το περιεχόμενο έχει φωτογραφίες, λίγα λόγια και επιλογές, δεν κουράζει. Το σημαντικό είναι να μην υπερφορτώνεται η οθόνη. Οι περισσότεροι θέλουν να πάρουν την πληροφορία, όχι να διαβάσουν κεφάλαιο.

Εικόνα 4.11: Εμφάνιση εκθεμάτων μιας συγκεκριμένης κατηγορίας

Στην αναζήτηση των εκθεμάτων υπάρχει η δυνατότητα επιλογής κατηγορίας εκθεμάτων, έτσι ώστε να ξέρει ο επισκέπτης τι ακριβώς ψάχνει και που πρέπει να το ψάξει. Όταν επιλέξει κατηγορία και ξεκινήσει την αναζήτηση, η εφαρμογή του εμφανίζει μόνο τα εκθέματα της επιλεγμένης κατηγορίας.

Η βασική λειτουργία αναζήτησης πραγματοποιείται με την παρακάτω μέθοδο:

```
Future<void> _searchExhibits(String query) async {
  final trimmed = query.trim();
  final searchTerm = trimmed.isEmpty ? '%' : '%$trimmed';
  _debounce?.cancel();
  _debounce = Timer(const Duration(milliseconds: 300), () async {
    final locale = Localizations.localeOf(context).languageCode;
```

```
      String searchColumn = locale == 'en' ? 'name_en' : 'name';
      final categoryId = categories[_selectedIndex]['id'];
      final shouldFilterCategory = categoryId != 'all';
      final response = await Supabase.instance.client
          .rpc('search_exhibits', params: {
        'search_term': searchTerm,
        'lang': searchColumn,
        'category_id': shouldFilterCategory ? categoryId : 'all',
      });
      final translated = <Map<String, dynamic>>[];
      print("🔍 Response: $response");

      for (var exhibit in response) {
        translated.add({
          "id": exhibit["id"] ?? "",
          "name": exhibit["name"] ?? "",
          "name_en": exhibit["name_en"] ?? "",
          "description": exhibit["description"] ?? "Δεν υπάρχει
περιγραφή.",
          "description_en": exhibit["description_en"] ?? "No
description.",
          "imageUrl": exhibit["imageUrl"] ?? "",
        });
        print("🔍 Response: ${exhibit['imageUrl']}");
      }
      if (mounted) {
        setState(() {
          searchResults = translated;
          isSearching = true;
        });
      }
    });
  }
```

Η συνάρτηση _searchExhibits είναι υπεύθυνη για την αναζήτηση εκθεμάτων με βάση τη λέξη-κλειδί που εισάγει ο χρήστης. Αρχικά, το κείμενο που πληκτρολογείται καθαρίζεται από περιττά κενά, και αν είναι άδειο, η αναζήτηση ορίζεται να επιστρέψει όλα τα αποτελέσματα (χρησιμοποιώντας τον χαρακτήρα % που σημαίνει «οτιδήποτε» σε SQL).

Για να αποφευχθεί η συνεχής αποστολή αιτημάτων προς τον διακομιστή όσο ο χρήστης πληκτρολογεί, εφαρμόζεται μια μικρή χρονική καθυστέρηση 300ms (μέσω μηχανισμού debounce). Μόλις περάσει αυτός ο χρόνος χωρίς νέα πληκτρολόγηση, γίνεται κλήση προς το backend (Supabase), το οποίο εκτελεί τη stored procedure search_exhibits.

Εικόνα 4.12: Παράδειγμα μη εύρεσης αποτελεσμάτων

Σε περίπτωση που πραγματοποιηθεί αναζήτηση και δεν βρεθούν εκθέματα που να περιέχουν στο όνομα τους αυτό που αναζητά ο χρήστης, τότε η εφαρμογή εμφανίζει κατάλληλο μήνυμα, όπως φαίνεται και στην παραπάνω εικόνα. Η αναζήτηση γίνεται σε διαφορετική στήλη ανάλογα με τη γλώσσα του χρήστη (π.χ. στα ελληνικά αναζητείται στη στήλη name, στα αγγλικά στη name_en).

Αν έχει επιλεγεί κάποια συγκεκριμένη κατηγορία, αυτή προστίθεται ως φίλτρο, διαφορετικά επιστρέφονται εκθέματα από όλες τις κατηγορίες.

Τα δεδομένα που επιστρέφονται από τη βάση μορφοποιούνται ώστε να περιλαμβάνουν μόνο τα απαραίτητα πεδία: αναγνωριστικό, όνομα, περιγραφή και εικόνα. Αν κάποια από αυτά λείπουν, το σύστημα τα αντικαθιστά με προεπιλεγμένες τιμές (π.χ. “Δεν υπάρχει περιγραφή”).

Τέλος, αν το γραφικό περιβάλλον είναι ακόμη ενεργό (ελέγχεται με το mounted), τα αποτελέσματα αποθηκεύονται και εμφανίζονται στον χρήστη, ενεργοποιώντας τη σχετική κατάσταση στην εφαρμογή.

Για τον σχεδιασμό τέτοιας εμπειρίας δεν απαιτείται πολύπλοκος εξοπλισμός. Ένα κινητό αρκεί. Η ευκολία είναι το κλειδί. Αν αρχίσουν να ζητούνται επιπλέον πράγματα, χάνεται το νόημα.

Αν στο μέλλον υπάρχει πρόθεση για επέκταση της εφαρμογής, μπορούν να προστεθούν λίγες επιλογές ακόμη. Για παράδειγμα, να αποθηκεύει ένας επισκέπτης τα εκθέματα που τον ενδιέφεραν. Ή να του δίνεται ένας σύνδεσμος να τα δει ξανά στο σπίτι. Όχι πολλά. Μόνο όσα χρειάζονται.

Καλό είναι επίσης η εφαρμογή να υποστηρίζει περισσότερες γλώσσες. Όχι απαραίτητα όλες από την αρχή, αλλά τουλάχιστον ελληνικά και αγγλικά. Και να αλλάζουν αυτόματα, χωρίς επιλογή. Απλώς να ακολουθούν τη γλώσσα του τηλεφώνου. Αυτό διευκολύνει χωρίς να μπερδεύει.

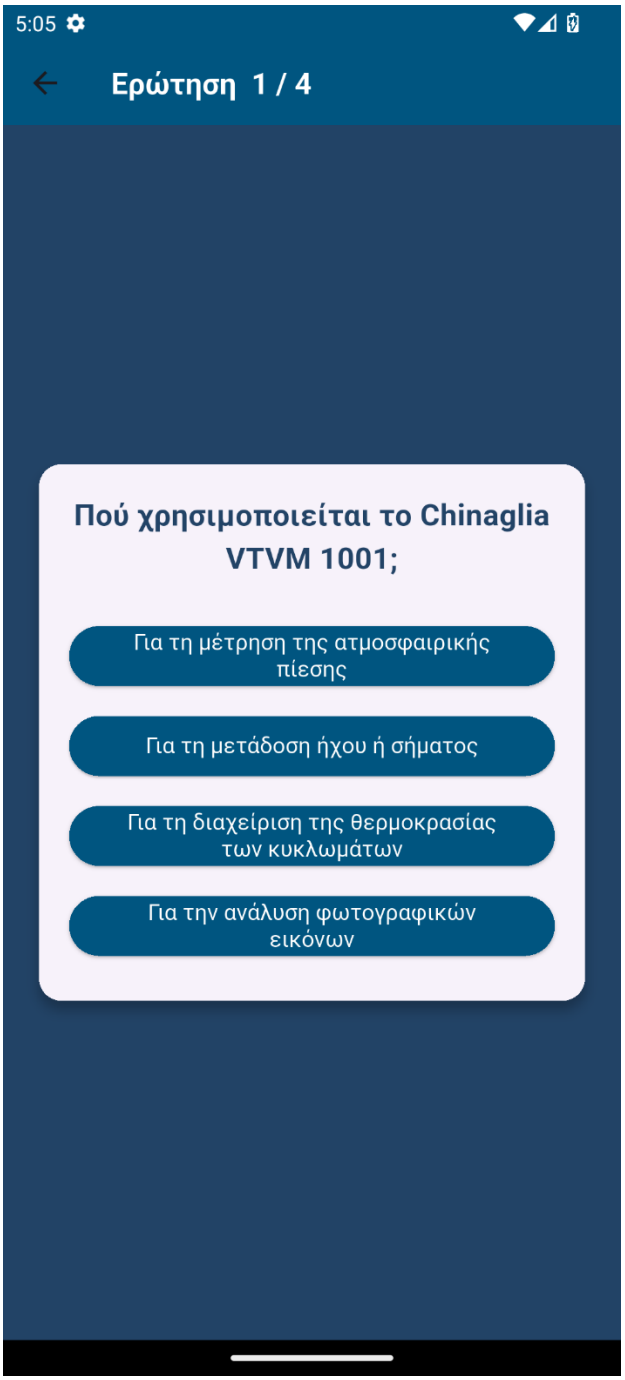
Το σημαντικό είναι να μην ενοχλεί. Να μην τραβάει την προσοχή από το ίδιο το μουσείο. Η τεχνολογία πρέπει να ακολουθεί. Όχι να προηγείται. Να βοηθάει όσους το θέλουν. Όσοι δεν το χρειάζονται, δεν πρέπει να αισθάνονται ότι χάνουν κάτι.

4.2.3 Κουίζ: Δομή & Δυναμική Αξιολόγηση

Η δραστηριότητα των ερωτήσεων σε μορφή κουίζ που σχεδιάστηκε για το Τεχνολογικό Μουσείο του ΔΙΠΑΕ δεν είχε σκοπό να αξιολογήσει ή να κρίνει τους επισκέπτες με κανέναν τρόπο. Αντίθετα, η λογική πίσω από τον σχεδιασμό βασίστηκε στην ιδέα ότι ο πιο άμεσος και ουσιαστικός τρόπος επικοινωνίας με το κοινό είναι το ίδιο το εκθεσιακό περιεχόμενο. Η προσέγγιση αυτή στηρίζεται στην πεποίθηση πως η εμπειρία του επισκέπτη διαμορφώνεται καλύτερα όταν του προσφέρεται η δυνατότητα να συνδεθεί με τα εκθέματα μέσα από την προσωπική του παρατήρηση και κατανόηση.

Κατά την περιήγηση, είναι φυσικό να προκύπτουν ερωτήματα για όσα παρουσιάζονται. Πολλές φορές, ο επισκέπτης παρατηρεί, εντοπίζει κάτι ενδιαφέρον, αλλά δεν προλαβαίνει ή δεν έχει τον χρόνο να εμβαθύνει. Με αυτό ως αφετηρία, σχεδιάστηκε ένα απλό κουίζ, βασισμένο σε πληροφορίες που αφορούν το αντίστοιχο έκθεμα. Οι ερωτήσεις δεν έχουν στόχο να ελέγξουν τι γνωρίζει ο επισκέπτης, αλλά να του δώσουν την ευκαιρία να σκεφτεί ξανά όσα είδε και να τα επεξεργαστεί με τον δικό του τρόπο. Αποτελούν τρόπο υπενθύμισης και ταυτόχρονα μια ευκαιρία να αναγνωρίσει ο επισκέπτης τι αποκόμισε και τι συγκράτησε από την εμπειρία του στον χώρο.

Με τη χρήση του κουίζ, δίνεται μια επιπλέον διάσταση στην επαφή με το περιεχόμενο. Δεν προστίθεται απλώς πληροφορία, αλλά ενισχύεται η σύνδεση ανάμεσα στη θέαση και τη μνήμη. Σε κάποιες περιπτώσεις, το κουίζ βοηθά να διαπιστωθεί αν οι βασικές έννοιες πέρασαν στον επισκέπτη, χωρίς να απαιτείται επίσημη αξιολόγηση. Το εργαλείο αυτό, τελικά, λειτουργεί υποστηρικτικά, εστιάζοντας όχι στη μέτρηση, αλλά στην ενίσχυση της μουσειακής εμπειρίας.



Εικόνα 4.13: Ερώτηση ενός εκθέματος

Η αρχική οθόνη του κουίζ δεν λειτουργεί αυτόνομα, ούτε εμφανίζεται τυχαία. Εντάσσεται μέσα στη συνολική ροή της εφαρμογής και δεν παρεμβαίνει στην εμπειρία του επισκέπτη. Δεν χρειάζεται να την αναζητηθεί από τον χρήστη εκ των προτέρων. Εμφανίζεται μόνο όταν έχει νόημα, δηλαδή αφού ο επισκέπτης έχει ήδη δει κάτι, έχει αλληλεπιδράσει, ή έχει σαρώσει έναν QR κωδικό που σχετίζεται με το περιεχόμενο του μουσείου.

Το κουίζ δεν παρουσιάζεται σαν υποχρέωση. Δεν υπάρχει κάποιο μήνυμα που να προτρέπει την διεξαγωγή του κουίζ από τον επισκέπτη. Είναι απλώς διαθέσιμο, σαν επιλογή. Αν κάποιος θέλει να το δοκιμάσει, το κάνει. Αν όχι, απλώς συνεχίζει με την περιήγηση.

Δεν υπάρχουν περιορισμοί στο ποιος μπορεί να το χρησιμοποιήσει. Δεν απαιτεί προηγούμενη γνώση ή προετοιμασία. Ούτε υπάρχει κάποιος σωστός τρόπος να το απαντήσει κανείς. Είναι περισσότερο ένας τρόπος να σκεφτεί κανείς τι πρόσεξε, τι θυμάται και τι του έκανε εντύπωση. Δεν έχει στόχο την αξιολόγηση, αλλά τη σύνδεση με την εμπειρία της επίσκεψης. Κάθε κουίζ του μουσείου είναι φτιαγμένο με βάση ένα συγκεκριμένο αντικείμενο ή μια θεματική ενότητα που αφορά το περιεχόμενο της έκθεσης. Αυτό σημαίνει πως μερικές ερωτήσεις που εμφανίζονται είναι γενικές και κάποιες άλλες προσαρμόζονται σε όσα έχουν προηγηθεί στην επίσκεψη. Είναι ερωτήσεις μικρές, με στόχο να βοηθήσουν και όχι να κουράσουν τον επισκέπτη. Παρουσιάζονται μαζί με τέσσερις επιλογές απαντήσεων και η μορφή τους είναι απλή: ο επισκέπτης βλέπει την ερώτηση, επιλέγει αυτό που θεωρεί σωστό και συνεχίζει. Δεν υπάρχουν περιττές ενέργειες ούτε κάτι που να αποσπά.

Η τεχνική λειτουργία πίσω από το κουίζ έχει σχεδιαστεί έτσι ώστε να επιτρέπει την αυτόματη φόρτωση των ερωτήσεων. Δηλαδή, δεν υπάρχουν προκαθορισμένες ερωτήσεις μέσα στην εφαρμογή από πριν. Όταν ο χρήστης ξεκινά ένα κουίζ, η εφαρμογή στέλνει ένα αίτημα στη βάση δεδομένων, ζητώντας να φορτωθούν οι σχετικές ερωτήσεις με βάση τον QR κωδικό που έχει ήδη σαρωθεί. Η διαδικασία βασίζεται στην παρακάτω εντολή:

```
final List<dynamic> response = await Supabase.instance.client
          .rpc('get_quiz_questions',params:{'qr_id':
widget.qrCode});
```

Η βάση κοιτάζει το πεδίο qr_id και επιστρέφει τις ερωτήσεις που έχουν συνδεθεί με αυτό το σημείο. Κάθε εγγραφή περιλαμβάνει την ίδια την ερώτηση, τις τέσσερις απαντήσεις, καθώς και ποια από αυτές είναι η σωστή. Με αυτόν τον τρόπο, το περιεχόμενο του κουίζ μπορεί να αλλάξει ανά πάσα στιγμή, χωρίς να χρειάζεται να βγει νέα έκδοση της εφαρμογής. Η πληροφορία αποθηκεύεται προσωρινά σε μια λίστα που κρατά τα δεδομένα σε μορφή List<dynamic>>.

Καθώς ο χρήστης απαντά σε κάθε ερώτηση, η εφαρμογή συγκρίνει την επιλογή του με τη σωστή απάντηση, μέσω της μεθόδου _checkAnswer. Αν υπάρχει συμφωνία, τότε καταγράφεται μία μονάδα στο σκορ του:

```
   void _checkAnswer(bool isCorrect) {
    if (isCorrect) {
      score++;
    }
    // Ελέγχουμε αν υπάρχει άλλη ερώτηση ή αν τελείωσε το Quiz
    if (currentQuestionIndex < questions.length - 1) {
      setState(() {
        currentQuestionIndex++;
      });
    } else {
```

```
      _showResults();
    }
  }
```

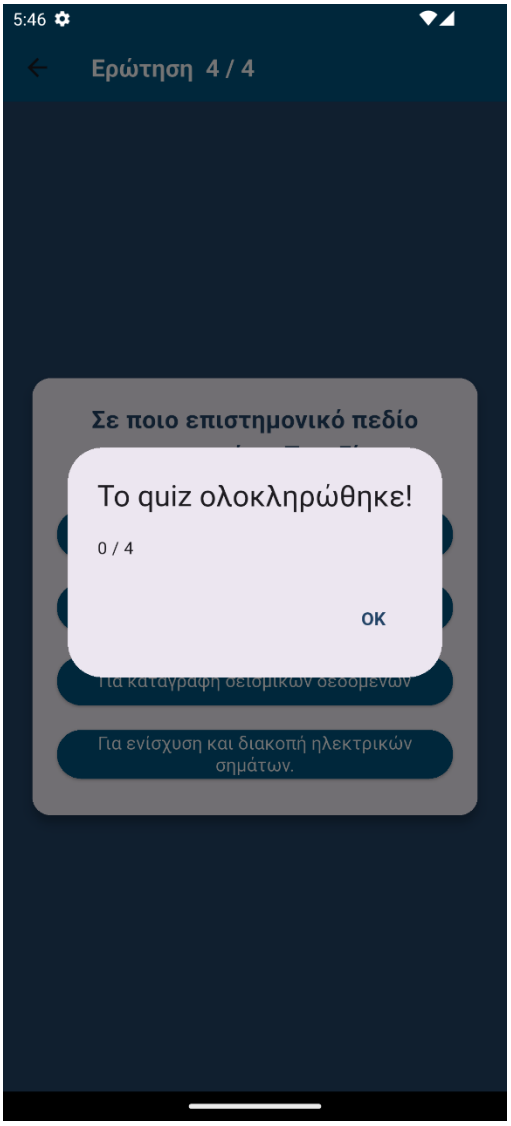
Μετά από κάθε απάντηση, η εφαρμογή κάνει μια μικρή παύση μερικών δευτερολέπτων και προχωρά είτε στην επόμενη ερώτηση είτε , αν έχουν απαντηθεί όλες , στην τελική οθόνη που δείχνει το αποτέλεσμα. Η αλλαγή οθόνης γίνεται με την εξής μέθοδο:

```
  void _showResults() {
    showDialog(
      context: context,
      builder: (context) {
        return AlertDialog(
          title: Text(AppLocalizations.of(context)!.quizComplete),
          content: Text("$score / ${questions.length}"),
          actions: [
            TextButton(
              onPressed: () {
                Navigator.pop(context);
                Navigator.pop(context);
              },
              child: const Text("OK"),
            ),
          ],
        );
      },
    );
  }
```

Η τελευταία οθόνη είναι πολύ απλή και καθαρή. Δείχνει τον συνολικό αριθμό σωστών απαντήσεων και συνοδεύεται από ένα φιλικό μήνυμα. Δεν γίνεται καμία σύγκριση με άλλους χρήστες, ούτε καταγράφεται το αποτέλεσμα. Δεν υπάρχει πίεση. Ο επισκέπτης μπορεί, αν το θελήσει, να επαναλάβει το κουίζ ή να το αφήσει και να συνεχίσει την περιήγησή του κανονικά, χωρίς καμία υποχρέωση.

Οι ερωτήσεις στο κουίζ παρουσιάζονται με τρόπο που δεν ενοχλεί το μάτι. Είναι καθαρές, χωρίς στολίδια, με απλό φόντο. Οι λέξεις είναι ευανάγνωστες και τα κουμπιά δεν είναι μεγάλα ή

φορτωμένα. Αντίθετα, είναι όσο πρέπει να είναι. Έχουν απόσταση μεταξύ τους για να μην συμβεί κατά λάθος πάτημα , κάτι που έχει τύχει σε άλλες εφαρμογές, όπου όλα είναι στρυμωγμένα. Δεν υπάρχουν κίνηση ή φώτα. Τίποτα που να τραβάει χωρίς λόγο την προσοχή. Το ζητούμενο δεν είναι να εντυπωσιαστεί κανείς, αλλά να μπορέσει να χρησιμοποιήσει αυτό το εργαλείο χωρίς να κουραστεί.



Εικόνα 4.14: Παράθυρο ολοκλήρωσης του κουίζ

Σε περίπτωση που ο επισκέπτης επαναλάβει το κουίζ για ένα έκθεμα, θα παρατηρήσει πως οι ερωτήσεις καθώς και οι απαντήσεις τους, εμφανίζονται με διαφορετική σειρά.. Έτσι, αν κάποιος ξαναμπεί στο κουίζ, υπάρχει πιθανότητα να δει ερωτήσεις με διαφορετική σειρά από πριν. Αυτό κρατάει το ενδιαφέρον. Δεν μοιάζει με κάτι που έχει ήδη γίνει. Ο επισκέπτης δεν βαριέται, και συνεχίζει χωρίς να νιώθει ότι απλώς επαναλαμβάνει.

Ο τρόπος που λειτουργεί η εφαρμογή δεν αλλάζει ανάλογα με το κινητό ή τη σύνδεση. Ακόμα κι αν το δίκτυο δεν είναι πολύ δυνατό, το κουίζ φορτώνεται. Δεν φαίνεται να χρειάζεται αναμονή. Δεν βαραίνει τη συσκευή, δεν προκαλεί διακοπές, και μπορεί να χρησιμοποιηθεί από πολλούς χρήστες στον ίδιο χώρο.

Επίσης υπάρχει η δυνατότητα ο επισκέπτης να πει τη γνώμη του για το μουσείο απαντώντας λίγες απλές ερωτήσεις, συμπληρώνοντας μια φόρμα. Αυτό μπορεί να το κάνει πατώντας το κουμπί 'Πες τη γνώμη σου' που βρίσκεται στο κάτω μέρος της αρχικής οθόνης της εφαρμογής.

Απ' την πλευρά του μουσείου, αυτό το σύστημα έχει και πρακτικό όφελος. Δεν υπάρχει ανάγκη να υπάρχει προσωπικό για να εξηγεί, ούτε κάποιος να επιβλέπει. Όλα γίνονται από το κινητό του επισκέπτη. Οι ερωτήσεις μπορούν να αλλάζουν, χωρίς να χρειαστεί να φτιαχτεί νέα έκδοση. Προσαρμόζεται. Το μουσείο απλώς αλλάζει τις ερωτήσεις από το σύστημα και το κουίζ συνεχίζει να λειτουργεί.

Τελικά, εκεί που βοηθά περισσότερο είναι στον ίδιο τον επισκέπτη. Δεν είναι θεατής. Συμμετέχει. Δεν του λένε μόνο τι να δει , κάνει κι εκείνος κάτι. Θυμάται, επιλέγει, σκέφτεται. Και όχι για να βρει το «σωστό». Δεν υπάρχει σωστό ή λάθος. Υπάρχει μια εμπειρία που, αν μη τι άλλο, του μένει λίγο περισσότερο.

4.2.4 Σύστημα Πολλαπλών Γλωσσών (Localization)

Όταν σχεδιάστηκε η εφαρμογή, μια από τις πρώτες σκέψεις που τέθηκαν ήταν το πώς θα τη χρησιμοποιούν άνθρωποι που δεν ξέρουν ελληνικά. Δεν είναι λίγες οι περιπτώσεις που μουσεία επισκέπτονται τουρίστες ή άνθρωποι από άλλες χώρες. Δεν είναι όλοι γνώστες της γλώσσας, κι αυτό δημιουργεί πρόβλημα στην κατανόηση. Οπότε, από την αρχή, υπήρχε η πρόθεση να υποστηρίζονται περισσότερες από μία γλώσσες.

Η λύση που επιλέχθηκε ήταν αρκετά απλή: η εφαρμογή εντοπίζει τη ρύθμιση γλώσσας του κινητού. Αν είναι στα ελληνικά, ξεκινάει την λειτουργία της στα ελληνικά. Αν είναι στα αγγλικά, τότε μεταφράζεται αυτόματα το περιβάλλον στην αγγλική γλώσσα. Δεν χρειάζεται δηλαδή ο χρήστης να πάει σε ρυθμίσεις ή να ψάξει κάποια επιλογή. Όλα γίνονται μόνα τους.

Αυτό γίνεται γιατί πίσω από κάθε φράση στην εφαρμογή υπάρχει ένα αρχείο που έχει γραμμένες τις λέξεις και τις αντίστοιχες μεταφράσεις τους. Δηλαδή, η εφαρμογή δεν γράφει απευθείας τα κουμπιά ή τα κείμενα. Αντί γι' αυτό, τραβάει κάθε φράση από ένα ειδικό αρχείο. Ανάλογα με τη γλώσσα του κινητού, παίρνει την ελληνική ή την αγγλική εκδοχή.

Για παράδειγμα, το "Ξεκινήστε το κουίζ" εμφανίζεται μέσα από τον κώδικα έτσι:

```
label: Text( AppLocalizations.of(context)!.startQuizButton,
            style: const TextStyle(color: Colors.white),)
```

Η λέξη startQuizButton συνδέεται με ένα αρχείο που έχει τις μεταφράσεις. Αν το κινητό είναι ρυθμισμένο στα ελληνικά, η εφαρμογή εμφανίζει το " Ξεκινήστε το κουίζ ". Αν είναι στα αγγλικά, τότε λέει "Start quiz". Αν στο μέλλον προστεθεί και τρίτη γλώσσα, π.χ. γαλλικά, θα προστεθεί μία γραμμή και για αυτή.

```
//app_en.arb
"startQuizButton": "Start Quiz",
```

```
//app_el.arb
"startQuizButton": "Ξεκινήστε το Quiz",
```

Η τεχνική αυτή βοηθάει πολύ γιατί έτσι, αν κάποιος χρειαστεί να αλλάξει κάτι, το κάνει σε ένα σημείο. Δεν χρειάζεται να περάσει από κάθε οθόνη της εφαρμογής. Όλα τα μεταφρασμένα κείμενα είναι συγκεντρωμένα στα λεγόμενα .arb αρχεία.

Το καλό είναι ότι αν μελλοντικά το μουσείο θελήσει να προσθέσει άλλη γλώσσα, για παράδειγμα γερμανικά ή ολλανδικά, το μόνο που χρειάζεται να γίνει είναι η μετάφραση των φράσεων. Όλος ο υπόλοιπος μηχανισμός παραμένει ίδιος.

Επίσης, όλα τα μηνύματα, ειδοποιήσεις και κουμπιά είναι ενταγμένα στο σύστημα αυτό. Δεν υπάρχουν σημεία που να μένουν στα ελληνικά αν η συσκευή είναι σε άλλη γλώσσα. Το περιβάλλον χρήσης είναι ενιαίο. Η αυτόματη γλώσσα είναι χρήσιμη και σε περιπτώσεις όπου ένα παιδί δανείζεται κινητό από κάποιον μεγαλύτερο, ή όταν το τηλέφωνο είναι σε κοινή χρήση.

Γενικά, το να βλέπει ο καθένας την εφαρμογή σε γλώσσα που καταλαβαίνει δεν είναι μόνο πρακτικό. Είναι και θέμα σεβασμού. Δεν είναι όλοι υποχρεωμένοι να μιλούν ελληνικά. Ούτε χρειάζεται να δυσκολευτούν για να καταλάβουν τι βλέπουν. Όταν τα πράγματα εμφανίζονται στη δική τους γλώσσα, νιώθουν πιο άνετα. Και αυτό είναι σημαντικό, ειδικά σε ένα περιβάλλον πολιτισμού.

Η προσθήκη γλωσσών ήταν από τα πράγματα που μπήκαν από νωρίς στον σχεδιασμό. Δεν προστέθηκε μετά. Ο λόγος ήταν απλός: όσο πιο φιλική είναι η εφαρμογή προς όλους, τόσο πιο πολλοί επισκέπτες, όπως φοιτητές από άλλες χώρες που έρχονται μέσω Erasmus, θα τη χρησιμοποιούν.

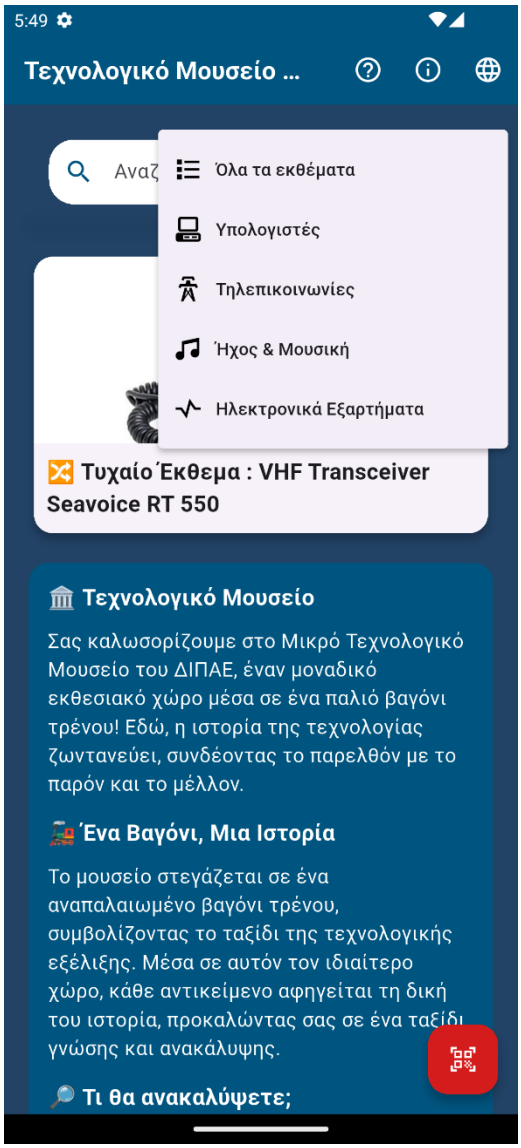
4.2.5 Εικονίδια, Αισθητική και Προσβασιμότητα

Όταν ένας επισκέπτης ανοίγει για πρώτη φορά μια εφαρμογή, δεν κοιτάζει πρώτα τον κώδικα, ούτε σκέφτεται πώς υλοποιήθηκε. Αυτό που τον ενδιαφέρει, χωρίς καν να το συνειδητοποιεί, είναι το πώς φαίνεται. Αν του φαίνεται φιλική, καθαρή, κατανοητή. Αν βρίσκει εύκολα αυτό που θέλει. Γι’ αυτό και ο τρόπος που έχουν σχεδιαστεί τα εικονίδια και το σύνολο της αισθητικής παίζουν πολύ σημαντικό ρόλο.

Από την αρχή, η σκέψη ήταν να μην κουράζει ο σχεδιασμός. Όλα τα στοιχεία, από τα κουμπιά μέχρι τα χρώματα, επιλέχθηκαν για να διευκολύνουν τη χρήση. Κανένα σημείο της οθόνης δεν υπάρχει απλώς για να είναι όμορφο. Όλα έχουν μια λειτουργία. Αυτό ισχύει και για τα εικονίδια που εμφανίζονται. Δεν είναι περίπλοκα ή εντυπωσιακά. Είναι απλά, καθαρά και το σημαντικότερο: θυμίζουν στον χρήστη κάτι γνώριμο. Το εικονίδιο για τη σάρωση μοιάζει με έναν QR, για το κουίζ υπάρχει ένα ερωτηματικό, και για τις πληροφορίες το γνωστό “i”.

Έγινε προσπάθεια να χρησιμοποιηθούν σχέδια που ο περισσότερος κόσμος έχει ξαναδεί. Όχι γιατί δεν θα μπορούσαν να γίνουν πιο μοντέρνα, αλλά γιατί δεν υπήρχε λόγος να μπλέξουμε. Ο στόχος ήταν να πατάει κάποιος ένα κουμπί επειδή καταλαβαίνει τι σημαίνει, όχι επειδή του το είπε κάποιος.

Σε όλα τα σημεία της εφαρμογής, αποφεύχθηκαν τα έντονα χρώματα που μπορεί να τραβήξουν την προσοχή άσκοπα. Το φόντο είναι ουδέτερο, κυρίως λευκό ή πολύ ανοιχτό, και τα υπόλοιπα στοιχεία ξεχωρίζουν αρκετά χωρίς να χτυπάνε στο μάτι. Τα κουμπιά είναι μεγάλα και έχουν ικανοποιητική απόσταση μεταξύ τους. Δεν στριμώχνονται και δεν χρειάζεται ακρίβεια για να τα πατήσει κανείς.



Εικόνα 4.15: Αρχική οθόνη. Διακρίνονται τα περισσότερα είδη εικονιδίων

Στην προσπάθεια να οργανωθεί η πλοήγηση και να δοθεί ξεκάθαρη εικόνα για τις θεματικές ενότητες των εκθεμάτων, χρησιμοποιήθηκαν εικονίδια από βιβλιοθήκες που είναι ήδη γνωστές στους περισσότερους σχεδιαστές, όπως τα Material Design Icons και το FontAwesome. Δεν επιλέχθηκαν τυχαία αυτά τα εικονίδια γιατί έχουν συγκεκριμένη αισθητική, είναι καθαρά, ευανάγνωστα και, το σημαντικότερο, φέρνουν στο μυαλό άμεσα την έννοια που αντιπροσωπεύουν. Για παράδειγμα, όπως φαίνεται και στην παραπάνω εικόνα ο υπολογιστής εμφανίζεται με το γνώριμο σχήμα του desktop που όλοι έχουμε δει κάπου, οι τηλεπικοινωνίες αποδίδονται με έναν πύργο εκπομπής , αρκετά συνηθισμένο , και η κατηγορία του ήχου και της μουσικής με ένα εικονίδιο που παραπέμπει σε νότες. Στην κατηγορία «όλα τα εκθέματα» χρησιμοποιήθηκε ένα είδος λίστας, κάτι σαν σημειωματάριο, ενώ για τα ηλεκτρονικά εξαρτήματα, που είναι πιο τεχνική κατηγορία, μπήκε το σύμβολο της αντίστασης, γνωστό από το χώρο της ηλεκτρολογίας. Το μενού με τα εικονίδια και τα ονόματα των κατηγοριών δημιουργούνται με την κλήση της παρακάτω μεθόδου:

```
Widget _buildCategoriesMenu(BuildContext context) {
  final locale = Localizations.localeOf(context).languageCode;
  return AnimatedContainer(
    duration: const Duration(milliseconds: 300),
    height: _showCategories ? 200 : 0,
    decoration: BoxDecoration(
      color: const Color(0xFF163E66),
      borderRadius: BorderRadius.circular(10),
      boxShadow: [
        BoxShadow(
          color: Colors.black.withOpacity(0.2),
          blurRadius: 10,
          spreadRadius: 2,
        ),
      ],
    ),
    child: SingleChildScrollView(
      child: Column(
        children: categories.map((category) => ListTile(
          leading: Icon(category['icon'], color: Colors.white),
          title: Text(
            locale=='en' ? category['name_en'] : category['name'],
            style: const TextStyle(color: Colors.white),
          ),
          onTap: () {
            setState(() {
              _selectedIndex = categories.indexOf(category);
              _showCategories = false;
            });
            _fetchExhibitsByCategory(category['id']);
          },
        )).toList(),
      ),
```

```
        ),
      );
    }
```

Το μενού των πτυσσόμενων κατηγοριών δημιουργήθηκε χρησιμοποιώντας το widget buildCategoriesMenu, το οποίο επιτρέπει στον χρήστη να επιλέξει οριζόντια την κατηγορία των εκθεμάτων που θέλει. Μέσω του AnimatedContainer δόθηκαν απαλές μεταβάσεις κατά την εμφάνιση ή την αφαίρεση του μενού ενώ παράλληλα συμβάλλεται σημαντικά στην ανάκαμψη της ταχύτητας της αλληλεπίδρασης. Έτσι, όταν ένας χρήστης επιλέξει μια κατηγορία, το μενού κλείνει αυτόματα και ξεκινά η επιστροφή του περιεχομένου με ομαλό τρόπο.

Σε επίπεδο βασικών λειτουργιών, οι επιλογές εικονιδίων βασίστηκαν σε καθιερωμένες αναφορές. Το σύμβολο της υδρογείου χρησιμοποιήθηκε για την αλλαγή γλώσσας , είναι το πιο διαδεδομένο εικονίδιο σε τέτοιες περιπτώσεις. Η αναζήτηση αποδόθηκε με τον κλασικό μεγεθυντικό φακό, η βοήθεια με το ερωτηματικό που βρίσκεται σχεδόν σε κάθε εφαρμογή και, τέλος, για τις πληροφορίες της εφαρμογής καθώς και του κουμπιού που οδηγεί στο GitHub προτιμήθηκε το κλασικό εικονίδιο πληροφοριών και το εικονίδιο της πλατφόρμας GitHub, αντίστοιχα.

Όλα αυτά εντάσσονται σε έναν γενικότερο σχεδιασμό που αποφεύγει τον εντυπωσιασμό. Δεν χρησιμοποιήθηκαν έντονα χρώματα, ούτε εφέ που αποσπούν την προσοχή χωρίς λόγο. Το φόντο παραμένει διακριτικό, έτσι ώστε το περιεχόμενο να προβάλλεται με σαφήνεια. Τα κουμπιά έχουν αρκετό χώρο μεταξύ τους και δεν στριμώχνονται. Ο χρήστης δεν χρειάζεται να στοχεύσει με ακρίβεια για να τα πατήσει, αρκεί μια φυσική κίνηση, κάτι που διευκολύνει ιδιαίτερα τη χρήση σε κινητές συσκευές.

Η γραμματοσειρά που χρησιμοποιήθηκε είναι σταθερή σε όλη την εφαρμογή. Ούτε αλλάζει μέγεθος από σελίδα σε σελίδα, ούτε χρώμα χωρίς λόγο. Είναι καθαρή, ευανάγνωστη, και δεν κουράζει. Δεν έχει περίεργα καλλιγραφικά στυλ, ούτε κάτι που θα μπορούσε να θεωρηθεί «φλύαρο» στον σχεδιασμό. Σε οθόνες μικρού μεγέθους, όπως σε πιο παλιά κινητά, όλα τα στοιχεία προσαρμόζονται ώστε να χωράνε καλά χωρίς να μικραίνουν υπερβολικά.

Η απλότητα του σχεδιασμού έχει βοηθήσει πολύ και στην προσβασιμότητα. Τα κουμπιά είναι τόσο μεγάλα όσο πρέπει, και τα χρώματα έχουν επιλεγεί έτσι ώστε να φαίνονται καθαρά ακόμα και από άτομα με περιορισμένη όραση ή δυσκολίες στην αναγνώριση χρωμάτων. Δεν υπάρχουν συνδυασμοί που θα μπέρδευαν κάποιον που έχει, για παράδειγμα, δυσχρωματοψία. Αυτό ήταν κάτι που λήφθηκε σοβαρά υπόψη.

Δεν χρησιμοποιήθηκαν εφέ που αλλάζουν την οθόνη ή κάνουν κινήσεις χωρίς λόγο. Η εφαρμογή είναι σταθερή. Ο στόχος ήταν να έχει κανείς τον έλεγχο της πλοήγησης, χωρίς να του αποσπάται η προσοχή. Για να το πούμε απλά, όλα γίνονται με έναν φυσικό τρόπο.

Επιπλέον, δεν χρειάζονται ειδικές χειρονομίες για να λειτουργήσει κάτι. Ούτε σάρωση με δύο δάχτυλα, ούτε σύρσιμο. Η μόνη περίπτωση που εφαρμόζεται κάτι τέτοιο, είναι όταν πρέπει να γίνει επαναφόρτωση της αρχικής οθόνης, η οποία πραγματοποιείται με ένα απλό σύρσιμο από πάνω προς τα κάτω. Όλα γίνονται με ένα απλό πάτημα. Αυτό βοηθά και παιδιά και ηλικιωμένους, αλλά και ανθρώπους που ίσως έχουν κάποια δυσκολία στο να χρησιμοποιήσουν πιο πολύπλοκες λειτουργίες. Ένα άγγιγμα αρκεί για κάθε ενέργεια.

Η προσβασιμότητα δεν αφορά μόνο τα χρώματα και το μέγεθος των γραμμάτων. Αφορά το πώς νιώθει ο χρήστης όταν χρησιμοποιεί την εφαρμογή. Αν έχει μπροστά του κάτι που του φαίνεται φιλικό, που δεν τον πιέζει, που τον αφήνει να προχωρήσει με τον δικό του ρυθμό, τότε είναι πολύ πιο πιθανό να ασχοληθεί πραγματικά.

Το σημαντικότερο είναι ότι η αισθητική αυτή σχεδιάστηκε για να βοηθήσει. Αν κάτι φαίνεται ωραίο, αλλά μπερδεύει τον χρήστη, δεν έχει θέση σε μια εφαρμογή σαν κι αυτή. Εδώ, κάθε χρώμα, κάθε κουμπί και κάθε απόσταση έχουν έναν απλό λόγο: να γίνει η χρήση πιο εύκολη.

4.3 Υλοποίηση Βάσης Δεδομένων με Supabase

Κατά την ανάπτυξη της εφαρμογής, η βάση δεδομένων δεν μπήκε απλώς για να εξυπηρετήσει τεχνικά. Ο ρόλος της ήταν πιο ουσιαστικός, δηλαδή πρέπει να οργανώνει το τι βλέπει ο επισκέπτης, πότε το βλέπει και με ποια σειρά. Ίσως δεν είναι άμεσα εμφανές, αλλά στην πράξη, χωρίς τη βάση, η εφαρμογή δεν έχει ροή.

Γι’ αυτό και η επιλογή της Supabase έγινε νωρίς. Όχι γιατί ήταν απλώς μοντέρνα λύση, αλλά γιατί προσέφερε κάτι που στην πράξη έλειπε. Λειτουργικό backend, χωρίς να απαιτείται να στηθεί εξ αρχής ολόκληρη υποδομή. Πρόκειται για πλατφόρμα BaaS (Backend as a Service), που χτίζεται πάνω σε PostgreSQL, κάτι που εγγυάται σταθερότητα, αλλά και μια κοινότητα από πίσω που έχει λύσει σχεδόν κάθε πιθανό πρόβλημα.

Αυτό που αποδείχθηκε καθοριστικό, όμως, ήταν το εξής: το περιβάλλον διαχείρισης είναι πολύ κοντά σε αυτό που ήδη ξέρουν πολλοί χρήστες, καθώς θυμίζει υπολογιστικό φύλλο. Δεν χρειάζεται γνώσεις προγραμματισμού. Αν κάποιος θέλει να αλλάξει έναν τίτλο, προσθέτει μια γραμμή. Αν χρειάζεται νέο έκθεμα, το καταχωρεί με τον QR και εμφανίζεται αυτόματα στην εφαρμογή. Χωρίς καμία τεχνική παρέμβαση.

Σε αυτήν την ενότητα, περιγράφεται πώς στήθηκε η βάση, τι πίνακες υπάρχουν, πώς λειτουργούν, και πώς επικοινωνούν με την εφαρμογή. Εξηγείται ακόμα και το πώς η εφαρμογή συλλέγει τα δεδομένα με κάθε σάρωση QR ή ενεργοποίηση κουίζ. Όλα αυτά, σε συνδυασμό με ένα βασικό μέτρο ασφάλειας, εξασφαλίζουν πως ο χρήστης δεν θα δει ποτέ κάτι λάθος ή κάτι που δεν θα έπρεπε να φαίνεται.

Η βασική λογική πίσω από την επιλογή της Supabase ήταν να μη χρειάζεται συνεχής τεχνική συντήρηση. Κι αυτό επετεύχθη. Δεν απαιτούνται σχεδόν καθόλου αλλαγές στον κώδικα για να προστεθεί ένα νέο έκθεμα. Η δομή της βάσης υποστηρίζει εύκολα ανανέωση, χωρίς επανεκκίνηση, χωρίς ενημέρωση της εφαρμογής, χωρίς εξαρτήσεις. Και αυτό είναι το πιο σημαντικό: μια λύση που δεν κουράζει κανέναν.

4.3.1 Βάση Δεδομένων: Πίνακες, Σχέσεις & Δεδομένα

Η βάση δεδομένων της εφαρμογής δεν είναι απλώς μια τοποθεσία όπου τοποθετούνται δεδομένα. Είναι η σπονδυλική στήλη του μουσείου μέσα στο ψηφιακό του περιβάλλον. Δεν αποθηκεύει απλώς πληροφορίες, αλλά συνδέει, ερμηνεύει και επαναφέρει περιεχόμενο τη στιγμή που χρειάζεται, ανάλογα με τη διαδρομή του κάθε επισκέπτη. Η αρχιτεκτονική βασίζεται στο Supabase, το οποίο από μόνο του βασίζεται στην PostgreSQL και δεν είναι μια επιλογή που έγινε τυχαία. Είναι μια τεχνολογία που στηρίζεται εδώ και χρόνια από κοινότητες και επαγγελματίες, κυρίως επειδή δεν δυσκολεύει και, αν χρειαστεί, επεκτείνεται χωρίς να ζητά καινούρια βάση. Η σύνδεση με την εφαρμογή γίνεται στο

παρασκήνιο, φέρνοντας στον επισκέπτη μόνο αυτό που του είναι χρήσιμο, όταν του είναι χρήσιμο. Η σύνδεση φαίνεται στο παρακάτω κομμάτι κώδικα:

```
void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  try {
    await dotenv.load(fileName: ".env");
  } catch (e) {
    print("❌ Σφάλμα φόρτωσης .env: $e");
  }
  await Supabase.initialize(
    url: dotenv.env['SUPABASE_URL']!,
    anonKey: dotenv.env['SUPABASE_ANON_KEY']!,
  );
  runApp(const MyApp());
}
```

Το συγκεκριμένο σημείο αποτελεί την αρχή της εφαρμογής και φροντίζει να ρυθμιστούν όλα όσα χρειάζονται πριν ξεκινήσει το περιβάλλον του Flutter. Αρχικά, καλείται το ensureInitialized(), για να διασφαλιστεί ότι το σύστημα είναι έτοιμο πριν τρέξουν ασύγχρονες λειτουργίες. Αμέσως μετά, επιχειρείται να φορτωθεί το αρχείο .env, από το οποίο αντλούνται βασικές παραμετροποιήσεις, όπως το URL και το ανώνυμο κλειδί για το Supabase. Το αρχείο .env είναι κρυφό γιατί έχει το URL και το ανώνυμο κλειδί και χρησιμοποιείται στον κώδικα για λόγους ασφαλείας (για να μην είναι εκτεθειμένες αυτές οι βασικές παράμετροι). Αν κάτι πάει στραβά εκεί, το σφάλμα απλώς καταγράφεται στο output χωρίς να σταματά η εφαρμογή. Μετά από αυτό, γίνεται η αρχικοποίηση του Supabase με τις παραμέτρους που μόλις φορτώθηκαν. Όταν όλα είναι έτοιμα, το runApp() εκκινεί την εφαρμογή κανονικά, φορτώνοντας το MyApp.

Στη συγκεκριμένη υλοποίηση έχουν χρησιμοποιηθεί δύο βασικοί πίνακες, και προς το παρόν αυτοί καλύπτουν τις ανάγκες. Ο ένας αφορά τα εκθέματα και οι πληροφορίες που τα συνοδεύουν. Ο άλλος αφορά τις ερωτήσεις των κουίζ που σχετίζονται με τα εκθέματα. Όλα τα υπόλοιπα , αποτελέσματα, και χρήστες , μπορεί να προστεθούν αν το σύστημα επεκταθεί, αλλά δεν υπάρχουν ακόμα στη βάση. Αυτό που υπάρχει είναι λειτουργικό και επαρκές.

Στον πίνακα valid_qr_codes είναι αποθηκευμένα τα εκθέματα. Ο κάθε QR κωδικός που βρίσκεται δίπλα ή πάνω σε ένα έκθεμα αντιστοιχεί στο αντίστοιχο πεδίο id σε αυτό τον πίνακα. Δεν μιλάμε μόνο για το όνομα, αλλά και για την περιγραφή, την κατηγορία και την εικόνα που το συνοδεύει. Όλα αυτά μαζί δίνουν το πλαίσιο στο οποίο θα κινηθεί η εμπειρία του επισκέπτη.

Πίνακας 4.1: Ο πίνακας valid_qr_codes

Πεδίο	Τύπος πεδίου	Περιγραφή
id	text	Ο μοναδικός αναγνωριστικός κωδικός, ίδιος με τον QR
name	text	Τίτλος εκθέματος στα ελληνικά
description	text	Περιγραφή του εκθέματος στα ελληνικά
description_en	text	Περιγραφή του εκθέματος στα αγγλικά
imageUrl	text	Σύνδεσμος προς την εικόνα του εκθέματος
name_en	text	Τίτλος εκθέματος στα αγγλικά
category	text	Κατηγορία του εκθέματος

Όταν ο επισκέπτης σαρώσει ένα QR που αντιστοιχεί σε κάποιο έκθεμα, η εφαρμογή αναζητά στον πίνακα valid_qr_codes και φορτώνει όλα τα πεδία του αντίστοιχου εκθέματος. Αν η γλώσσα είναι τα αγγλικά, τότε τα name_en και description_en φορτώνονται αντί για τα πεδία name και description.

Στον πίνακας quizzes είναι αποθηκευμένες οι ερωτήσεις, που είναι το άλλο βασικό στοιχείο. Για κάθε έκθεμα υπάρχουν τέσσερις ερωτήσεις. Η σύνδεση γίνεται ξανά μέσω του id, δηλαδή του ίδιου κωδικού που έχει και το έκθεμα. Με αυτόν τον τρόπο, όταν ο επισκέπτης φτάσει στο κουίζ, βλέπει μόνο τις ερωτήσεις που σχετίζονται με αυτό που μόλις είδε.

Πίνακας 4.2: Πίνακας quizzes

Πεδίο	Τύπος πεδίου	Περιγραφή
id	text	Κωδικός εκθέματος
question	text	Κείμενο της ερώτησης
answers	jsonb	JSON με τις πιθανές απαντήσεις
uid	uuid	αναγνωριστικό ερώτησης

Παράδειγμα του πεδίου answers σε μορφή JSON:

```
[{"text": "Για την ενίσχυση μικροκυμάτων",
     "correct": false},
{"text": "Για πλοήγηση και εντοπισμό",
"correct": true},
```

```
{"text": "Για τη διαχείριση της θερμοκρασίας των κυκλωμάτων",
"correct": false},
{"text": "Για καταγραφή σεισμικών δεδομένων",
"correct": false}]
```

Αυτός ο τρόπος επιτρέπει στην εφαρμογή να προβάλλει εύκολα τις επιλογές στον χρήστη, χωρίς να χρειάζεται περίπλοκες δομές ή πολλαπλά πεδία. Αν η εφαρμογή χρειαστεί αργότερα περισσότερες επιλογές ή ερωτήσεις, το σχήμα του πίνακα δεν αλλάζει.

Η βάση δεν στηρίζεται μόνο σε απλή ανάγνωση δεδομένων, αλλά και σε προκαθορισμένες συναρτήσεις (functions), οι οποίες διευκολύνουν την πρόσβαση, την αναζήτηση καθώς και την ασφάλεια. Αυτές δεν είναι απλώς φίλτρα. Είναι μικρές «λογικές μονάδες» που εκτελούν συγκεκριμένα αιτήματα, συνήθως με βάση τα στοιχεία που παρέχει ο χρήστης μέσα από την εφαρμογή.

Ακολουθούν οι βασικές functions που υπάρχουν στη βάση:

Πίνακας 4.3: Αποθηκευμένες διαδικασίες της βάσης

Όνομα συνάρτησης	Περιγραφή / Ρόλος
get_exhibits_by_category(category_input text)	Φέρνει όλα τα εκθέματα μιας συγκεκριμένης κατηγορίας από τον πίνακα valid_qr_codes
get_qr_code_info(qr_id text)	Δίνει τις λεπτομέρειες του εκθέματος με βάση το QR που σαρώθηκε
get_quiz_questions(qr_id text)	Επιστρέφει τις ερωτήσεις του κουίζ που σχετίζονται με συγκεκριμένο έκθεμα
get_random_exhibit()	Φέρνει τυχαία ένα έκθεμα (για την αρχική οθόνη)
get_random_quiz_questions(limit_count integer)	Επιλέγει τυχαίες ερωτήσεις, μέχρι έναν αριθμό
search_exhibits(search_term text, lang text, category text)	Επιτρέπει αναζήτηση εκθεμάτων βάσει λέξης-κλειδιού, γλώσσας και κατηγορίας

Οι συναρτήσεις αυτές δεν χρειάζονται ειδική πρόσβαση (ασφάλεια τύπου "Invoker") και είναι σχεδιασμένες έτσι ώστε να αξιοποιούνται απευθείας από την εφαρμογή.

Αυτός είναι προς το παρόν ο πλήρης κορμός της βάσης. Ούτε υπερβολές, ούτε σπατάλη δομών. Ό,τι υπάρχει εξυπηρετεί έναν σκοπό κι αν χρειαστεί να επεκταθεί, η υπάρχουσα διάταξη το επιτρέπει.

4.3.2 Ροή δεδομένων μέσα στην εφαρμογή (ερωτήματα, απαντήσεις, πλοήγηση)

Η ροή δεδομένων σε μια εφαρμογή σαν αυτή δεν είναι κάτι κρυφό ή περίπλοκο, αλλά ούτε και κάτι που μένει στο παρασκήνιο χωρίς σημασία. Αντίθετα, είναι αυτή που δίνει "ζωή" στο σύστημα και εξηγεί στον επισκέπτη πού βρίσκεται και τι μπορεί να κάνει. Κάθε του ενέργεια ξεκινάει από μια αφορμή, κι αυτή, σχεδόν πάντα, είναι η σάρωση ενός QR.

Ο χρήστης στέκεται μπροστά σε ένα έκθεμα, εντοπίζει το μικρό τετράγωνο αυτοκόλλητο, σηκώνει το κινητό του και σαρώνει οποιονδήποτε κωδικό QR. Από εκείνη τη στιγμή και μετά, η εφαρμογή αναλαμβάνει. Το πρώτο πράγμα που κάνει είναι να επικοινωνήσει με τη βάση , χωρίς το παραμικρό βήμα παραπάνω. Κάθε QR κωδικός, όπως έχει σχεδιαστεί, έχει αντιστοιχηθεί με συγκεκριμένο id στη Supabase. Με το που εντοπιστεί, ενεργοποιείται μια συνάρτηση: get_qr_code_info.

```
try {
      final List<dynamic> result = await Supabase.instance.client
          .rpc('get_qr_code_info', params: {'qr_id': code});
      final response = result.isNotEmpty ? result.first : null;
      if (response != null) {
        Navigator.pushReplacement(
          context,
          MaterialPageRoute(
            builder: (context) => QRInfoScreen(
              id: response['id'] ?? '',
              name: response['name'] ?? '',
              name_en: response['name_en'] ?? '',
              description: response['description'] ?? '',
              description_en: response['description_en'] ?? '',
              imageUrl: response['imageUrl'] ?? '',
            ),
          ),
        );
      } else {
        if (!_hasShownInvalidQrMessage) {
          _hasShownInvalidQrMessage = true;
          ScaffoldMessenger.of(context).showSnackBar(
            SnackBar(
              content: Text(
                AppLocalizations.of(context)!.invalidQrMessage,
                style:    const    TextStyle(color:    Colors.white,
fontWeight: FontWeight.bold),
              ),
              backgroundColor: Colors.red,
```

```
              duration: const Duration(seconds: 3),
            ),
          );
          Future.delayed(const Duration(seconds: 3), () {
            _hasShownInvalidQrMessage = false;
          });
        }
      }
    }
```

Αυτό το κομμάτι κώδικα έχει να κάνει με την ανάγνωση ενός QR κωδικού και την εμφάνιση των σχετικών πληροφοριών. Αν η συνάρτηση επιστρέψει κάποιο αποτέλεσμα, δηλαδή αν ο QR αντιστοιχεί σε κάποιο έκθεμα, τότε ο χρήστης μεταφέρεται αυτόματα σε μια νέα οθόνη όπου βλέπει πληροφορίες όπως όνομα, περιγραφή και εικόνα του αντικειμένου. Αυτά τα στοιχεία είναι και στα ελληνικά και στα αγγλικά, ανάλογα με τις ρυθμίσεις της εφαρμογής.

Αν όμως δεν βρεθεί τίποτα, δηλαδή αν ο QR δεν είναι καταχωρημένος, εμφανίζεται στην οθόνη ένα απλό μήνυμα που λέει ότι ο κωδικός δεν είναι έγκυρος. Για να μην κουράζεται ο χρήστης, το μήνυμα εμφανίζεται μόνο μία φορά και φεύγει μετά από λίγα δευτερόλεπτα. Έτσι αποφεύγονται τα ενοχλητικά επαναλαμβανόμενα μηνύματα.

Ο σκοπός είναι να γίνεται ο χειρισμός είτε βρεθεί αποτέλεσμα είτε όχι, χωρίς να μπερδευτεί ο χρήστης ή να διακοπεί η εμπειρία του μέσα στην εφαρμογή.

Δεν ζητιέται κάτι υπερβολικό , απλώς να επιστραφούν τα βασικά. Τίτλος, περιγραφή, εικόνα, κατηγορία. Τα απολύτως αναγκαία για να καταλάβει ο επισκέπτης τι έχει μπροστά του και γιατί του έχει τραβήξει την προσοχή. Όλα αυτά τραβιούνται σε μια κλήση. Η εφαρμογή δεν περιμένει πολύ. Η πληροφορία εμφανίζεται σχεδόν αμέσως.

Από εκεί και πέρα, υπάρχει και κουίζ που συνοδεύει το κάθε έκθεμα. Όχι για να καθυστερήσει τον χρήστη, αλλά για να του προτείνει κάτι παραπάνω. Μια δεύτερη συνάρτηση, η get_quiz_questions, ελέγχει αν για το συγκεκριμένο qr_id έχουν δηλωθεί ερωτήσεις.

Όλο αυτό λειτουργεί στο παρασκήνιο και δεν ζητά από τον χρήστη να κάνει τίποτα παραπάνω απ’ το να κοιτάξει και να επιλέξει. Η πλοήγηση, με τον τρόπο που έχει οργανωθεί, δεν κουράζει και δεν μπερδεύει. Αν θέλει να γυρίσει πίσω, το κάνει με το κουμπί της συσκευής ή με το πλήκτρο στο πάνω μέρος της εφαρμογής. Αν θέλει να δει κάτι άλλο, μπορεί να το κάνει χωρίς να κλείσει την εφαρμογή.

```
try {
      final List<dynamic> response = await Supabase.instance.client
          .rpc('get_quiz_questions', params: {'qr_id':
widget.qrCode});
      if (response.isNotEmpty) {
```

```
        final translated = await Future.wait(response.map((question) async {
          final originalQ = question['question'];
          final answers = List<Map<String, dynamic>>.from(question['answers']);
          String translatedQ = originalQ;
          if (locale == 'en') {
            translatedQ = await TranslationHelper.translate(originalQ, 'el', 'en');
            for (var answer in answers) {
              answer['text'] = await TranslationHelper.translate(answer['text'], 'el', 'en');
            }
          }
          answers.shuffle();
          return {
            'question': translatedQ,
            'answers': answers,
          };
        }));
        setState(() {
          questions = List<Map<String, dynamic>>.from(translated);
          isLoading = false;
        });
      } else {
        setState(() => isLoading = false);
      }
    } catch (e) {
      print("❌ Σφάλμα φόρτωσης ερωτήσεων: $e");
      setState(() => isLoading = false);
    }
```

Το συγκεκριμένο κομμάτι κώδικα έχει να κάνει με τη φόρτωση των ερωτήσεων για το κουίζ. Η εφαρμογή στέλνει το qr_id στη βάση μέσω μιας αποθηκευμένης συνάρτησης που λέγεται get_quiz_questions. Αν επιστραφούν αποτελέσματα, αυτά επεξεργάζονται και εμφανίζονται στην

οθόνη. Για κάθε ερώτηση που επιστρέφεται, η εφαρμογή παίρνει το κείμενο και τις απαντήσεις. Αν ο χρήστης έχει την εφαρμογή στα αγγλικά, τότε γίνεται μετάφραση των ερωτήσεων και των απαντήσεων από ελληνικά σε αγγλικά. Αυτό γίνεται με μια βοηθητική κλάση TranslationHelper που έχει γραφτεί γι' αυτό τον σκοπό. Οι απαντήσεις μάλιστα ανακατεύονται (shuffle), για να μην εμφανίζονται πάντα με την ίδια σειρά. Μόλις είναι όλα έτοιμα, η λίστα με τις ερωτήσεις αποθηκεύεται στη μεταβλητή questions και η ένδειξη φόρτωσης (isLoading) απενεργοποιείται, ώστε να εμφανιστεί το περιεχόμενο. Αν δεν επιστραφεί τίποτα από τη βάση ή αν υπάρξει κάποιο σφάλμα, η εφαρμογή απλώς σταματά τη φόρτωση χωρίς να σταματάει απότομα τη λειτουργία της. Έτσι, ακόμα και αν κάτι πάει στραβά, ο χρήστης δεν βλέπει λάθος ή κολλημένη οθόνη, αλλά κατάλληλο μήνυμα.

Υπάρχει και μια ακόμα δυνατότητα, λιγότερο προβεβλημένη αλλά χρήσιμη: να δει κανείς ένα τυχαίο έκθεμα. Γι’ αυτό υπάρχει η συνάρτηση get_random_exhibit. Δεν χρειάζεται να κάνει αναζήτηση. Μπορεί απλώς να πατήσει και να βρεθεί μπροστά σε κάτι καινούργιο, κάτι που δεν περίμενε.

```
final response = await Supabase.instance.client
          .rpc('get_random_exhibit')
          .maybeSingle();
```

Αυτή η γραμμή κώδικα ζητά από τη βάση να επιστρέψει ένα τυχαίο έκθεμα. Η συνάρτηση get_random_exhibit είναι αποθηκευμένη στη βάση (ως stored procedure) και καλείται μέσω Supabase. Το .maybeSingle() χρησιμοποιείται για να επιστρέψει ένα μόνο αποτέλεσμα, αν υπάρχει. Αν δεν βρεθεί τίποτα, δεν εμφανίζεται λάθος — απλώς το αποτέλεσμα είναι null.

Με αυτόν τον τρόπο, η εφαρμογή μπορεί να τραβάει ένα έκθεμα στην τύχη. Είναι χρήσιμο για λειτουργίες τύπου “δοκίμασε κάτι καινούριο” ή απλώς για να δείχνει κάτι τυχαίο στην αρχική οθόνη. Το σημαντικό είναι ότι γίνεται χωρίς να σταματάει να λειτουργεί η εφαρμογή, ακόμα κι αν δεν υπάρχουν δεδομένα.

Η σειρά, όπως δουλεύει στην πράξη, είναι η εξής:

- Ο επισκέπτης σαρώνει έναν QR.
- Η εφαρμογή παίρνει το qr_id και ψάχνει πληροφορίες με τη get_qr_code_info.
- Αν τις βρει, τις προβάλλει με τρόπο καθαρό.
- Αν ο χρήστης επιλέξει το κουίζ, τότε ενεργοποιείται και η get_quiz_questions.
- Αν βρεθούν ερωτήσεις, εμφανίζονται οι ερωτήσεις για να ξεκινήσει το κουίζ.
- Ο χρήστης απαντά ή όχι , χωρίς να δεσμεύεται για τίποτα.
- Σε κάθε σημείο μπορεί να επιστρέψει, να σαρώσει ξανά ή να προχωρήσει αλλού.

Δεν υπάρχει πίεση, δεν υπάρχει κρυφή λογική. Ο στόχος ήταν από την αρχή η εφαρμογή να "στέκεται δίπλα" στον επισκέπτη και να τον βοηθά μόνο όταν το ζητήσει.

Ακόμα κι όταν όλα τα παραπάνω κρύβονται πίσω από κώδικα, η αίσθηση πρέπει να είναι απλή. Να μην του φαίνεται "υπολογιστικό" όλο αυτό, αλλά ανθρώπινο. Η βάση κάνει τη δουλειά της, κι ο χρήστης απλώς κοιτάζει, διαβάζει, μαθαίνει ή παίζει. Ό,τι θέλει.

4.3.3 Πολιτικές Πρόσβασης (Row-Level Security)

Όσο λιτή κι αν είναι μια εφαρμογή, όταν επικοινωνεί με μια απομακρυσμένη βάση, η συζήτηση για ασφάλεια δεν είναι ποτέ περιττή. Στην παρούσα περίπτωση, η Supabase προσφέρει μια σταθερή βάση που υποστηρίζει αυθεντικοποίηση, διαχωρισμό ρόλων και , ίσως το πιο σημαντικό , Row-Level Security, το οποίο είναι ένας από τους πιο ευέλικτους τρόπους για να περιοριστεί η πρόσβαση στα δεδομένα.

Η συγκεκριμένη εφαρμογή, από τη φύση της, δεν απαιτεί σύνδεση χρήστη. Όλες οι πληροφορίες είναι διαθέσιμες σε κάθε επισκέπτη με την ίδια μορφή. Δεν υπάρχουν προσωπικά δεδομένα, ούτε ανάγκη προσωποποιημένων αποτελεσμάτων. Αυτή η απόφαση, που ήταν εξαρχής συνειδητή, απλοποιεί το μοντέλο χρήσης και μειώνει δραστικά τους πιθανούς κινδύνους διαρροής ή κακής διαχείρισης πληροφορίας.

Ωστόσο, αυτό δεν σημαίνει πως η εφαρμογή είναι “ανοικτή” ή ότι οποιοσδήποτε μπορεί να κάνει ό,τι θέλει. Αντιθέτως, όλη η ροή στηρίζεται σε καλά ελεγχόμενες functions. Δηλαδή, δεν εκτελούνται άμεσα queries προς τους πίνακες, αλλά μόνο μέσω προκαθορισμένων διαδρομών , όπως:

- get_qr_code_info(qr_id)
- get_quiz_questions(qr_id)
- get_exhibits_by_category(category)
- search_exhibits(...)

Αυτό σημαίνει ότι ακόμα και αν κάποιος προσπαθούσε να “καλέσει” απευθείας τα δεδομένα εκτός εφαρμογής, δεν θα μπορούσε να το κάνει εύκολα. Οι συναρτήσεις εκτελούνται με συγκεκριμένα όρια, και ορίζονται με το καθεστώς SECURITY INVOKER, πράγμα που σημαίνει ότι τρέχουν με τα δικαιώματα του χρήστη που τις κάλεσε , όχι με πλήρη πρόσβαση. Αυτό το στοιχείο, αν και τεχνικό, είναι ουσιαστικό: προφυλάσσει τη βάση από το να αποκαλύψει πληροφορίες που δεν πρέπει.

Αξίζει επίσης να αναφερθεί πως, ακόμη και με αυτή την απλοποιημένη ροή, μπορούν να ενεργοποιηθούν ανά πάσα στιγμή κανόνες ασφαλείας ανά γραμμή (row-level). Για παράδειγμα, αν στο μέλλον προστεθεί σύστημα αυθεντικοποίησης, μπορεί πολύ εύκολα να εφαρμοστεί κανόνας τύπου:

```
CREATE POLICY "Only allow access to user's own data"
ON results
FOR SELECT
USING (auth.uid() = user_id);
```

Αυτό σημαίνει πως κάθε χρήστης θα βλέπει μόνο τα δικά του αποτελέσματα , τίποτα παραπάνω.

Ακόμα κι αν κάτι τέτοιο δεν είναι προς το παρόν απαραίτητο, η βάση είναι ήδη δομημένη έτσι ώστε να το υποστηρίξει χωρίς επανεγκατάσταση ή ριζική αλλαγή. Η υποδομή υπάρχει , αρκεί μια γραμμή στον πίνακα και ένας κανόνας στον πίνακα πολιτικής για να περιοριστεί η πρόσβαση όπου χρειάζεται.

Ειδικά για functions που χρησιμοποιούν αναζήτηση (search_exhibits) ή επιστροφή τυχαίου εκθέματος (get_random_exhibit), τα δεδομένα περνούν μόνο από επιλεγμένα πεδία. Δεν επιστρέφονται, για παράδειγμα, metadata, timestamps, IDs δημιουργών ή εσωτερικά στοιχεία της βάσης. Αυτό κρατάει τα πράγματα καθαρά και περιορίζει την "έκθεση" της δομής σε εξωτερικό χρήστη.

Με λίγα λόγια:

- Όλα τα δεδομένα διανέμονται μόνο μέσω functions.
- Οι functions είναι περιορισμένες ως προς το τι επιστρέφουν.
- Δεν υπάρχει σύνδεση χρήστη, άρα δεν απαιτούνται policies αυτή τη στιγμή.
- Η υποδομή για RLS υπάρχει και μπορεί να ενεργοποιηθεί με μία εντολή.

Αν προστεθεί στο μέλλον κάποιο dashboard για διαχειριστές ή κάποιος πίνακας με αποτελέσματα χρηστών, τότε τα ίδια εργαλεία (Row-Level Security και Supabase Auth) μπορούν να αναλάβουν χωρίς νέο σχεδιασμό.

Μπορεί όλα αυτά να μη φαίνονται στον χρήστη που κοιτάει την οθόνη του, αλλά στο παρασκήνιο δημιουργούν ένα σύστημα σταθερό, ήσυχο, και , κυρίως , έτοιμο για το παραπάνω βήμα.

4.3.4 Συντήρηση και ανανέωση περιεχομένου

Η εφαρμογή δεν σχεδιάστηκε για να μείνει στάσιμη. Από την αρχή, ο στόχος ήταν να μπορεί να προσαρμόζεται εύκολα στις συνθήκες και στις αλλαγές που μπορεί να προκύψουν στο μουσείο. Αυτό σημαίνει πως το περιεχόμενό της, δηλαδή τα εκθέματα, οι πληροφορίες και τα κουίζ, πρέπει να μπορούν να αλλάζουν χωρίς να χρειάζεται κάποιος να ξέρει προγραμματισμό ή να έχει τεχνικές γνώσεις.

Η λογική πίσω από τον τρόπο που στήθηκε βασίστηκε στην ιδέα ότι την εφαρμογή δεν θα τη διαχειρίζεται κάποιος developer, αλλά άτομα που ασχολούνται με το περιεχόμενο. Για τον λόγο αυτό, επιλέχθηκαν εργαλεία που είναι απλά και κατανοητά. Το Supabase αποδείχθηκε πολύ χρήσιμο σ' αυτό. Το περιβάλλον του θυμίζει πίνακα ή υπολογιστικό φύλλο — δεν διαφέρει πολύ από ένα Excel. Μπορεί κάποιος να αλλάξει την περιγραφή ενός εκθέματος, να προσθέσει έναν νέο τίτλο ή να δηλώσει έναν QR κωδικό, και η εφαρμογή το ενημερώνει αυτόματα, χωρίς καμία επιπλέον ενέργεια.

Το ίδιο ισχύει και για τα κουίζ. Ο τρόπος που είναι φτιαγμένοι οι πίνακες είναι απλός: κάθε σειρά έχει μια ερώτηση, με τις πιθανές απαντήσεις και τη σωστή. Αν κάποιος θέλει να προσθέσει μια νέα ερώτηση ή να διορθώσει μία υπάρχουσα, το κάνει εύκολα, όπως θα έγραφε ένα email ή μια σημείωση. Δεν χρειάζεται να γνωρίζει πώς δουλεύει το σύστημα στο παρασκήνιο. Αρκεί να δηλώσει τον σωστό qr_id για να γίνει η σύνδεση με το σωστό έκθεμα — όλα τα υπόλοιπα τα αναλαμβάνει η εφαρμογή.

Η διαδικασία της ανανέωσης είναι άμεση και δεν δημιουργεί καθυστερήσεις. Αν ένα καινούργιο έκθεμα τοποθετηθεί στον χώρο και προστεθεί στη βάση μαζί με τον QR του, μπορεί να εμφανιστεί στην εφαρμογή μέσα σε λίγα λεπτά. Από εκείνη τη στιγμή, όποιος έχει εγκατεστημένη την εφαρμογή και σκανάρει τον κωδικό, θα δει το περιεχόμενο κανονικά, χωρίς να χρειάζεται να κάνει αναβάθμιση ή να κατεβάσει κάτι από το Play Store.

Αυτό είναι ίσως και το βασικότερο στοιχείο του συστήματος: η εφαρμογή παραμένει ζωντανή. Δεν στηρίζεται σε παλιές πληροφορίες και δεν χρειάζεται τεχνική επέμβαση για να παρακολουθεί τις αλλαγές. Εξελίσσεται μαζί με το μουσείο και το υλικό του. Είναι χρήσιμη όχι μόνο για τους επισκέπτες, αλλά και για όσους θέλουν να κρατήσουν το μουσείο ενεργό, σύγχρονο και προσιτό.

4.3.5 Συμπεράσματα , ρόλος της βάσης στην εμπειρία χρήστη

Αυτό που φαίνεται στην οθόνη, είτε είναι τίτλος, εικόνα, ερώτηση ή απάντηση, δεν είναι απλά περιεχόμενο. Πίσω του υπάρχει μια ολόκληρη διαδρομή που ξεκινάει από τη βάση δεδομένων. Στην περίπτωση της συγκεκριμένης εφαρμογής, η βάση δεν λειτουργεί απλώς ως "αποθήκη" για να κρατάει τα δεδομένα. Είναι ενεργό κομμάτι της εμπειρίας. Εκεί αποθηκεύονται οι πληροφορίες, και από εκεί ξεκινάει η διαδικασία με την οποία τελικά ο χρήστης βλέπει και αλληλεπιδρά με το περιβάλλον. Η βάση είναι αυτή που καθορίζει τι θα εμφανιστεί, τι θα τραβήξει την προσοχή, τι θα μείνει στον επισκέπτη. Δεν πρόκειται για μια τεχνική λεπτομέρεια , είναι το θεμέλιο πάνω στο οποίο πατάει όλη η εφαρμογή.

Η επιλογή του Supabase δεν έγινε επειδή ήταν η "μοντέρνα λύση", ούτε επειδή ήταν κάτι υποχρεωτικό. Επιλέχθηκε επειδή ταίριαξε πρακτικά με τις ανάγκες της εφαρμογής. Ήταν γρήγορη στην απόκριση, είχε ξεκάθαρη λογική και δεν έφερνε μαζί της περιττή πολυπλοκότητα. Το σημαντικότερο όμως είναι ότι επιτρέπει τη διαχείριση του περιεχομένου από ανθρώπους που γνωρίζουν τον χώρο του μουσείου και δεν χρειάζεται να έχουν τεχνικές γνώσεις. Δεν εξαρτάται δηλαδή από προγραμματιστές. Το περιβάλλον της θυμίζει πίνακα, σαν ένα spreadsheet. Οποιοσδήποτε μπορεί να αλλάξει μια περιγραφή, να προσθέσει έναν νέο QR ή να διορθώσει κάτι, χωρίς να μπει σε τεχνικές λεπτομέρειες. Και αυτό γίνεται άμεσα, χωρίς να χρειάζεται ενημέρωση ή επαναφορά της εφαρμογής.

Το ίδιο ισχύει και για τα κουίζ. Οι ερωτήσεις και οι απαντήσεις είναι οργανωμένες σε απλούς πίνακες. Κάθε γραμμή περιέχει μια ερώτηση, τις πιθανές επιλογές και τη σωστή απάντηση. Αν κάποιος θέλει να προσθέσει κάτι, το κάνει όπως θα συμπλήρωνε μια φόρμα ή ένα απλό κείμενο. Δεν χρειάζεται να γνωρίζει ούτε JSON, ούτε να καταλαβαίνει αν κάτι «συνδέεται» σωστά. Αρκεί να βάλει τον κατάλληλο qr_id, και η εφαρμογή φροντίζει για τα υπόλοιπα.

Η ανανέωση γίνεται χωρίς διαδικασίες. Αν μπει ένα νέο έκθεμα στο μουσείο και του αντιστοιχηθεί QR, μπορεί να προστεθεί στη βάση μέσα σε λίγα λεπτά. Την επόμενη φορά που κάποιος θα σαρώσει τον κωδικό, το περιεχόμενο θα εμφανιστεί αυτόματα. Δεν χρειάζεται να γίνει update στην εφαρμογή.

Αυτό είναι τελικά και το πιο σημαντικό. Η εφαρμογή μένει ζωντανή. Δεν είναι στατική, ούτε περιορίζεται σε περιεχόμενο που εντάχθηκε μια φορά. Μπορεί να αλλάζει, να εμπλουτίζεται και να εξελίσσεται μαζί με το μουσείο. Δεν είναι εργαλείο μόνο για τον επισκέπτη, αλλά και για όσους εργάζονται στον χώρο και θέλουν να κρατούν την εμπειρία ενδιαφέρουσα.

4.4 Ασφάλεια, Περιορισμοί και Δοκιμές

Από την πρώτη στιγμή, η λέξη "ασφάλεια" δεν εμφανίστηκε σαν πρόβλημα προς επίλυση. Εμφανίστηκε σαν ανάγκη που πηγάζει φυσικά, όταν θέλεις μια εφαρμογή να σέβεται τον χρήστη και να λειτουργεί απλά. Δεν υπήρχε σκοπός να δημιουργηθεί ένα εργαλείο που καταγράφει, αποθηκεύει ή αναλύει συμπεριφορές. Αντίθετα, η πρόθεση ήταν να μείνει η χρήση όσο πιο διακριτική γίνεται.

Η συγκεκριμένη εφαρμογή, η οποία δημιουργήθηκε για χρήση μέσα σε ένα μικρό τεχνολογικό μουσείο, δεν ζητά εγγραφές, δεν έχει λογαριασμούς, δεν αναγνωρίζει πρόσωπα. Το βασικό πράγμα που κάνει είναι να εμφανίζει πληροφορίες όταν κάποιος σαρώνει έναν QR κωδικό.

Δεν χρειάστηκε να προβλεφθούν μηχανισμοί προστασίας προσωπικών δεδομένων, γιατί δεν υπήρχαν προσωπικά δεδομένα. Δεν υπήρχε ανάγκη για terms & conditions σε πέντε παραγράφους, γιατί η χρήση είναι άμεση, χωρίς δέσμευση. Η εφαρμογή ανοίγει, ο επισκέπτης σκανάρει έναν κωδικό, και βλέπει. Τίποτα παραπάνω.

Η ίδια η βάση δεδομένων, που φιλοξενείται στη Supabase, είναι διαμορφωμένη έτσι ώστε να λειτουργεί σχεδόν μονόπλευρα. Επιτρέπει “διάβασμα” των πληροφοριών που έχουν εισαχθεί , αλλά όχι τροποποιήσεις από τον τελικό χρήστη. Ακόμα κι αν κάποιος προσπαθούσε να παρέμβει, δεν θα μπορούσε να αλλάξει τίποτα. Δεν υπάρχει καν η δυνατότητα από τον frontend κώδικα να σταλεί εντολή που πειράζει τα δεδομένα. Δεν γράφεται, δεν αποθηκεύεται τίποτα από την πλευρά του χρήστη και αυτό είναι απόλυτα συνειδητό.

Για την προσβασιμότητα των δεδομένων, χρησιμοποιούνται ασφαλή tokens τα οποία δημιουργούνται μέσα από τη Supabase και δεν αποθηκεύονται καθόλου στο κινητό του χρήστη. Ούτε καν προσωρινά. Κάθε αίτημα γίνεται μέσω ενός προκαθορισμένου καναλιού επικοινωνίας.

Όλα τα QR που γίνεται η σάρωση, ελέγχονται ως προς την εγκυρότητα. Αν δεν αντιστοιχεί σε υπάρχον κωδικό, η εφαρμογή απλώς ενημερώνει ότι “δεν βρέθηκε περιεχόμενο”. Δεν δείχνει τεχνικό μήνυμα, δεν σπάει, δεν παγώνει. Και αυτό, σε επίπεδο εμπειρίας, είναι ίσως εξίσου σημαντικό με την ασφάλεια καθαυτή: το σύστημα διαχειρίζεται τα λάθη με ευγένεια.

Στο επίπεδο των περιορισμών, υπήρξαν επίσης συνειδητές επιλογές. Δεν υποστηρίχθηκε χρήση offline. Για να λειτουργήσει η εφαρμογή, χρειάζεται σύνδεση στο διαδίκτυο. Το σκεπτικό εδώ ήταν απλό: εφόσον τα εκθέματα δεν αλλάζουν διαρκώς, δεν χρειάζεται να “κατέβουν” όλα τα δεδομένα στο κινητό. Αν γίνει μια αλλαγή στο περιεχόμενο, πρέπει να φανεί άμεσα σε όλους. Αυτό μπορεί να το εγγυηθεί μόνο η σύνδεση με τη Supabase σε πραγματικό χρόνο. Οπότε, η offline λειτουργία απορρίφθηκε.

Αν κάποιος χρήστης προσπαθήσει να σαρώσει QR χωρίς σύνδεση, εμφανίζεται καθαρό μήνυμα: “Απαιτείται σύνδεση στο διαδίκτυο”. Ούτε κολλάει η εφαρμογή, ούτε απορρίπτεται η ενέργεια χωρίς λόγο. Απλώς ενημερώνεται. Και εδώ φαίνεται ξανά ότι η εμπειρία χρήστη είχε προτεραιότητα , ακόμα και σε συνθήκες αποτυχίας.

Το γεγονός ότι δεν υπάρχουν εγγεγραμμένοι χρήστες περιορίζει μεν τις λειτουργίες, αλλά ανοίγει χώρο για μεγαλύτερη ασφάλεια. Αν δεν υπάρχει ταυτότητα, δεν υπάρχει λόγος προστασίας. Και από τη στιγμή που η χρήση είναι ανώνυμη, ο καθένας έχει την ίδια εμπειρία. Δεν υπάρχει διάκριση, δεν υπάρχει παρακολούθηση, δεν υπάρχει αποθήκευση “συμπεριφορών”.

Δεν είναι όμως όλα θέμα περιορισμών. Υπάρχει και το κομμάτι των εσωτερικών ελέγχων. Για παράδειγμα, οι διαχειριστές μπορούν να προσθέτουν ή να αφαιρούν εκθέματα μέσα από τη βάση. Αυτό γίνεται μόνο με login στον λογαριασμό Supabase και φυσικά, μόνο από εξουσιοδοτημένα άτομα. Οι πίνακες είναι προστατευμένοι με πολιτικές που δεν επιτρέπουν πρόσβαση από μη εξουσιοδοτημένους χρήστες, ακόμα κι αν κάποιος βρει το URL της βάσης.

Επίσης, έγινε μέριμνα ώστε καμία λειτουργία που εμφανίζεται στην εφαρμογή να μην είναι “ενεργή” αν δεν υποστηρίζεται πλήρως. Αν, για παράδειγμα, δεν υπάρχουν κουίζ για ένα συγκεκριμένο έκθεμα, η εφαρμογή ενημερώνει πως δεν υπάρχουν ερωτήσεις. Δεν οδηγεί σε κενές σελίδες. Αυτός είναι κι ένας άλλος τρόπος “ασφάλειας”: να μη φαίνεται τίποτα που δεν είναι έτοιμο.

Όλες οι διαδρομές πλοήγησης μέσα στην εφαρμογή ελέγχονται με τρόπο που αποτρέπει τις ακατάστατες μεταβάσεις. Δηλαδή, δεν μπορεί ο χρήστης να “κολλήσει” κάπου.

Αν θέλει να επιστρέψει, υπάρχει κουμπί. Αν δεν βρει περιεχόμενο, γυρίζει πίσω με μία κίνηση. Δεν υπάρχουν αδιέξοδα.

Η ασφάλεια, λοιπόν, δεν αντιμετωπίστηκε ως ένας τομέας ξεχωριστός από την εμπειρία χρήσης. Το αντίθετο: ενσωματώθηκε στην ίδια τη λογική του σχεδιασμού.

4.5 Συνολική Εμπειρία Χρήστη & Ανάλυση Ανάπτυξης

Δεν υπήρξε εξαρχής σχέδιο με βήματα και οδηγίες. Ούτε κάποιο πρότυπο. Μόνο η σκέψη πως, σε έναν χώρο όπως το μουσείο, η τεχνολογία μπορεί να έχει ρόλο , αλλά ήσυχο. Όχι με σκοπό να κλέψει την προσοχή. Περισσότερο για να σταθεί δίπλα στην εμπειρία, να συμπληρώσει ό,τι ίσως χάνεται, χωρίς να το παραμορφώσει.

Κάπως έτσι υλοποιήθηκε η εφαρμογή. Σαν εργαλείο, καθαρό και απλό. Ούτε για να εντυπωσιάσει ούτε για να πρωταγωνιστήσει. Αλλά για να βοηθήσει. Για να είναι εκεί, όταν και μόνο όταν χρειάζεται.

Η αρχή έγινε λιτά. Δυο-τρεις βασικές οθόνες, ένα κουμπί για σάρωση QR, μια σελίδα πληροφοριών και μια απλή δομή quiz. Τίποτα παραπάνω. Ή τουλάχιστον, όχι τότε. Το ζητούμενο δεν ήταν να χτιστεί κάτι μεγάλο. Ήταν να φανεί αν αυτό το λίγο λειτουργεί. Αν στέκει μέσα στον χώρο. Αν δένει.

Το Flutter επιλέχθηκε γιατί επέτρεπε να δοκιμάζεις γρήγορα. Να αλλάξεις κάτι και να το δεις αμέσως στην οθόνη. Δεν ήταν θέμα “τεχνολογικής υπεροχής”. Ήταν απλώς πρακτικό. Το ίδιο και το Supabase: ούτε φανταχτερό, ούτε υπερβολικό. Αλλά ακριβώς αυτό που χρειαζόταν. Κάτι σταθερό, που να συνδέεται εύκολα με την εφαρμογή, χωρίς να απαιτεί περίπλοκη διαχείριση.

Στην πορεία, πολλά πράγματα άλλαξαν. Άλλα αφαιρέθηκαν εντελώς. Για παράδειγμα, στην αρχή γινόταν φόρτωση όλων των quiz από την αρχή. Γρήγορα φάνηκε ότι αυτό δεν είχε νόημα. Δεν θα τα έβλεπε ποτέ όλα ο χρήστης. Έγινε δυναμική φόρτωση, μόνο όταν χρειάζεται. Ένα απλό παράδειγμα, αλλά χαρακτηριστικό. Η ίδια λογική εφαρμόστηκε παντού.

Κάποιες φορές, το πρόβλημα δεν ήταν ο κώδικας. Ήταν η χρήση. Όταν, για παράδειγμα, κάποιος προσπαθούσε να σκανάρει σε σκοτεινό σημείο, η κάμερα δεν ανταποκρινόταν με την ίδια ευκολία και έτσι προστέθηκε η επιλογή της ενεργοποίησης του φακού. Δεν ήταν στο αρχικό πλάνο, αλλά μπήκε γιατί προέκυψε από την πράξη. Όπως πολλά άλλα.

Το πιο σημαντικό κομμάτι τελικά δεν ήταν η τεχνολογία. Ήταν η αίσθηση. Το ότι, όταν κάποιος πατάει κάπου, γίνεται αυτό που περίμενε. Τίποτα παραπάνω , και τίποτα λιγότερο. Δεν υπήρχαν σφάλματα γιατί δεν υπήρχαν πολύπλοκα σενάρια. Το κάθε στοιχείο έμπαινε για κάποιον λόγο και όταν έπαυε να εξυπηρετεί, έβγαινε.

Η βάση δεδομένων στήθηκε λιτά. Πίνακες για κωδικούς QR, ερωτήσεις quiz, και γλωσσικές μεταφράσεις. Τίποτα περίπλοκο. Δεν χρειαζόταν. Ούτε αναλύσεις συμπεριφοράς, ούτε μεγάλα datasets. Το μόνο που μετρούσε ήταν η ροή: σάρωση, προβολή εκθεμάτων, προαιρετική απάντηση στις ερωτήσεις του κουίζ και συνέχεια στην πλοήγηση.

Κι αυτή η λογική, τελικά, μπορεί να σταθεί κι αλλού. Σε σχολεία, διαδρομές, εκδηλώσεις. Οπουδήποτε χρειάζεται κάτι μικρό, καθαρό, χωρίς πολλές οδηγίες. Η τεχνολογία δεν χρειάζεται να πρωταγωνιστεί για να είναι χρήσιμη. Μπορεί να μείνει στο πλάι , κι όμως, να κάνει τη διαφορά.

Κεφάλαιο 5ο: Συμπεράσματα ή/και βελτιώσεις

Το ξεκίνημα της διπλωματικής εργασίας και στην αρχή της ανάπτυξης της εφαρμογής δεν ήταν αποτέλεσμα κάποιας συστηματικής μελέτης ή οργανωμένου σχεδίου. Αντίθετα, τα πρώτα βήματα έγιναν μάλλον αυθόρμητα, με οδηγό μια γενική αίσθηση ότι υπήρχε ανάγκη για κάτι απλό, λειτουργικό και διακριτικό. Η ιδέα δεν ήταν να δημιουργηθεί μία εφαρμογή που θα ξεχωρίζει και δεν θα επιβάλλεται στον χρήστη. Σκοπός ήταν να υπάρξει ένα εργαλείο που να εντάσσεται ήσυχα μέσα στον χώρο του μουσείου, χωρίς να διαταράσσει την εμπειρία της επίσκεψης.

Η εφαρμογή σχεδιάστηκε με στόχο να λειτουργεί «παράλληλα» με τον επισκέπτη. Δεν αποσκοπούσε στο να του επιβληθεί ή να του υποδείξει κάτι, αλλά να είναι εκεί για να υποστηρίξει, αν και όταν χρειαστεί. Ήταν εξαρχής σαφές ότι η έμφαση δεν θα δινόταν σε γραφικά ή εφέ, αλλά στην καθαρή, απλή λειτουργικότητα. Ούτε περίπλοκα μενού, ούτε αχρείαστες επιλογές αλλά μόνο τα απολύτως απαραίτητα. Αυτή η εστίαση στο «λιγότερο» είχε ως στόχο να μειώσει τον κίνδυνο αποτυχίας και να ενισχύσει τη σταθερότητα.

Από πολύ νωρίς υιοθετήθηκε η λογική πως, αν κάτι δεν προσφέρει ουσιαστική αξία, καλύτερα να λείπει. Έτσι, αποφεύχθηκε η συσσώρευση λειτουργιών που θα μπορούσαν να μπερδέψουν ή να κουράσουν τον χρήστη. Αντί για συνεχή υπενθύμιση της παρουσίας της, η εφαρμογή έχει σχεδιαστεί ώστε να «υποχωρεί» όταν δεν είναι αναγκαία.

Τα τεχνολογικά εργαλεία που επιλέχθηκαν εξυπηρετούσαν αυτή την κατεύθυνση. Το Flutter επέτρεψε γρήγορη ανανέωση και πειραματισμούς χωρίς μεγάλες καθυστερήσεις στην εκτέλεση. Το Supabase κάλυψε τις ανάγκες της βάσης δεδομένων χωρίς να απαιτείται σύνθετη υποδομή ή περίπλοκες εξαρτήσεις. Δύο απλοί πίνακες, ένας για τα εκθέματα και ένας για τα κουίζ, ήταν αρκετοί για τις βασικές λειτουργίες.

Η δοκιμή στον πραγματικό χώρο αποκάλυψε όσα η θεωρία δεν μπορούσε να προβλέψει. Ο επισκέπτης του μουσείου δεν πηγαίνει εκεί για να ασχοληθεί με μια εφαρμογή. Αν δεν καταλαβαίνει άμεσα πώς λειτουργεί, το πιθανότερο είναι να την αγνοήσει. Αυτό έγινε ξεκάθαρο από τις πρώτες κιόλας φορές που δοκιμάστηκε η εφαρμογή επιτόπου.

Προέκυψαν ζητήματα σε σημεία που δεν είχαν υπολογιστεί. Για παράδειγμα, σε περιοχές με αμυδρό φωτισμό, τα κινητά δεν μπορούσαν εύκολα να αναγνωρίσουν τον QR κωδικό. Η προσθήκη της δυνατότητας ενεργοποίησης του φακού ήταν μία από τις αλλαγές που προέκυψαν με βάση την εμπειρία και όχι τον αρχικό σχεδιασμό. Αυτή η μικρή παρέμβαση αποδείχθηκε χρήσιμη. Στην πραγματικότητα, πολλές αποφάσεις πάρθηκαν με αυτόν τον τρόπο: όχι θεωρητικά, αλλά μέσα από δοκιμή και αναπροσαρμογή.

Το αποτέλεσμα δεν εντυπωσιάζει με την πρώτη ματιά, ούτε αυτός ήταν ο σκοπός. Αντίθετα, αυτό που προσφέρει είναι σταθερότητα και μια σαφής πρόθεση να μην διακόπτει την εμπειρία του επισκέπτη. Δεν αποσπά την προσοχή. Δεν καθοδηγεί. Είναι παρούσα όταν χρειάζεται και αόρατη όταν δεν χρειάζεται.

Η δομή της εφαρμογής επιτρέπει στο μέλλον να προστεθούν λειτουργίες χωρίς να διαταραχθεί η ισορροπία. Θα μπορούσε, για παράδειγμα, να ενσωματωθεί ένα απλό εργαλείο παρακολούθησης του αριθμού των επισκεπτών του μουσείου, όχι με στόχο την ανάλυση των χρηστών, αλλά για να παρέχεται ανατροφοδότηση στην ομάδα του μουσείου. Επιπλέον, θα μπορούσε να προβλεφθεί η

δυνατότητα διαχείρισης περιεχομένου από μη τεχνικά πρόσωπα, κάτι σημαντικό για χώρους όπου το επιστημονικό προσωπικό έχει καλύτερη γνώση των εκθεμάτων από ό,τι των εργαλείων πληροφορικής.

Μια πιθανή μελλοντική κατεύθυνση αφορά την ενσωμάτωση τεχνολογιών επαυξημένης πραγματικότητας (AR), επιτρέποντας στον επισκέπτη να δει το έκθεμα όπως ήταν κατά την περίοδο χρήσης του. Παράλληλα, η αρχιτεκτονική επιτρέπει την εύκολη προσθήκη νέων γλωσσών, βασικής offline λειτουργίας για περιοχές χωρίς επαρκές σήμα, ακόμη και ακουστικής περιήγησης για επισκέπτες που δυσκολεύονται στην ανάγνωση.

Συνολικά, η αξία της εφαρμογής δεν έγκειται στην τεχνολογική της πολυπλοκότητα, αλλά στην προσέγγιση που την ορίζει. Όλα σχεδιάστηκαν με γνώμονα τον σεβασμό προς τον χρήστη. Δεν επιδιώχθηκε ποτέ να εντυπωσιάσει και το ζητούμενο ήταν να υποστηρίξει την εμπειρία του μουσείου με τρόπο ουσιαστικό αλλά αθόρυβο. Αν τελικά λειτουργεί χωρίς να γίνεται αισθητή, τότε ίσως αυτό να είναι και η πιο ουσιαστική της επιτυχία.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] S. Tiwari, "An Introduction to QR Code Technology," 2016 International Conference on Information Technology (ICIT), Bhubaneswar, India, 2016, pp. 39-44, doi: 10.1109/ICIT.2016.021

[2] “Contextual QR codes,” IEEE Conference Publication | IEEE Xplore, Jul. 01, 2008. https://ieeexplore.ieee.org/abstract/document/4591344

[3] Peter Kieseberg, Manuel Leithner, Martin Mulazzani, Lindsay Munroe, Sebastian Schrittwieser, Mayank Sinha, and Edgar Weippl. 2010. QR code security. In Proceedings of the 8th International Conference on Advances in Mobile Computing and Multimedia (MoMM '10). Association for Computing Machinery, New York, NY, USA, 430,435. https://doi.org/10.1145/1971519.1971593

[4] Singh, S. (2016). QR code analysis. International Journal of Advanced Research in Computer Science and Software Engineering, 6(5).

[5] Liu, Yue, Ju Yang, and Mingjun Liu. "Recognition of QR Code with mobile phones." 2008 Chinese control and decision conference. IEEE, 2008.

[6] Rijksmuseum, “App , Rijksmuseum,” [Online]. Διαθέσιμο: https://www.rijksmuseum.nl/en/whats-on/app.

[7] Museums + Heritage Advisor, “British Museum reveals new app for self-guided tours,” *Museums + Heritage Advisor*, 24 May 2023. [Online]. Διαθέσιμο: https://museumsandheritage.com/advisor/posts/british-museum-reveals-new-app-for-self-guided-tours/.

[8] American Museum of Natural History, “Explorer , AMNH NYC,” [Online]. Διαθέσιμο: https://apps.apple.com/us/app/explorer-amnh-nyc/id381227123.

[9] American Museum of Natural History, “Explorer App Press Release,” Nov. 2016. [Online]. Διαθέσιμο: https://www.amnh.org/content/download/152456/2540763/file/explorer-app-press-release-final2.pdf.

[10] Museumfy, “Museumfy , Digital Museum Guide,” [Online]. Διαθέσιμο: https://museumfy.com.

[11] Musée du Louvre, “Welcome to the Louvre,” [Online]. Διαθέσιμο: https://www.louvre.fr/en.

[12] Smithsonian Institution, “Smithsonian Announces New Visitors Guide Mobile App,” [Online]. Διαθέσιμο: https://www.si.edu/newsdesk/releases/smithsonian-announces-new-visitors-guide-mobile-app.

[13] The Metropolitan Museum of Art, “The New Met App,” [Online]. Διαθέσιμο: https://www.metmuseum.org/perspectives/the-new-met-app.

[14] Vatican Museums, “Vatican Museums , Official Website,” [Online]. Διαθέσιμο: https://www.museivaticani.va.

[15] Android Developers. (n.d.). *Android Studio overview*. https://developer.android.com/studio

[16] Vogella, L. (n.d.). *Android Studio tutorial*. Vogella. https://www.vogella.com/tutorials/Android/article.html

[17] JetBrains. (n.d.). “*IntelliJ Platform SDK*.” [Online]. Διαθέσιμο: https://plugins.jetbrains.com/docs/intellij/welcome.html

[18] Stack Overflow. (2023). *Developer survey 2023*. [Online]. Διαθέσιμο: https://survey.stackoverflow.co/2023/

[19] Google, “Flutter: Build apps for any screen,” [Online]. Διαθέσιμο: https://flutter.dev

[20] Dart Team, “Dart FAQ,” *dart.dev*, 2024. [Online]. Διαθέσιμο: https://dart.dev/resources/faq#can-i-compile-dart-code-to-native-code
[21] Flutter Team, “Hot reload,” *Flutter Documentation*, 2024. [Online]. Διαθέσιμο: https://docs.flutter.dev/tools/hot-reload
[22] Flutter Team, “Building layouts,” *Flutter Documentation*, 2024. [Online]. Διαθέσιμο: https://docs.flutter.dev/ui/layout
[23] Flutter Team, “Technical overview: Rendering engine,” *Flutter Documentation*, 2024. [Online]. Διαθέσιμο: https://docs.flutter.dev/resources/technical-overview#rendering-engine
[24] Flutter.dev. “Get started - Install on Windows.” Διαθέσιμο: https://docs.flutter.dev/get-started/install/windows
[25] Flutter.dev. “flutter doctor.” Διαθέσιμο: https://docs.flutter.dev/reference/flutter-cli#flutter-doctor
[26] React Native, “React Native · A framework for building native apps using React, 2024. [Online]. Διαθέσιμο: https://reactnative.dev/
[27] Supabase, “What is Supabase,” Supabase Docs, 2024. [Online]. Διαθέσιμο: https://supabase.com/docs
[28] Supabase, “Architecture | Supabase Docs, 2024. [Online]. Διαθέσιμο: https://supabase.com/docs/guides/getting-started/architecture

ΠΑΡΑΡΤΗΜΑ Α : ΚΩΔΙΚΑΣ ΕΦΑΡΜΟΓΗΣ

Στο συγκεκριμένο παράρτημα παρουσιάζεται ο κώδικας της εφαρμογής, όπως αυτός συντάχθηκε στο Android Studio.

Main.dart

```
import 'package:connectivity_plus/connectivity_plus.dart';
import 'package:flutter/foundation.dart';
import 'dart:async';
import 'dart:io';
import 'dart:ui' as ui;
import 'dart:math';
import 'package:flutter/material.dart';
import 'package:flutter_dotenv/flutter_dotenv.dart';
import 'package:font_awesome_flutter/font_awesome_flutter.dart';
import
'package:material_design_icons_flutter/material_design_icons_flutter.dart';
import 'package:shared_preferences/shared_preferences.dart';
import 'package:supabase_flutter/supabase_flutter.dart';
import 'final_quiz_screen.dart';
import 'qr_info_screen.dart';
import 'qr_scanner_screen.dart';
import 'package:url_launcher/url_launcher.dart';
import 'package:flutter_localizations/flutter_localizations.dart';
import 'package:flutter_gen/gen_l10n/app_localizations.dart';


void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  try {
    await dotenv.load(fileName: ".env");
  } catch (e) {
    print("❌ Σφάλμα φόρτωσης .env: $e");
  }
  await Supabase.initialize(
    url: dotenv.env['SUPABASE_URL']!,
    anonKey: dotenv.env['SUPABASE_ANON_KEY']!,
  );
```

```
  runApp(const MyApp());
}

class MyApp extends StatefulWidget {
  const MyApp({super.key});

  static void setLocale(BuildContext context, Locale newLocale) {
    final _MyAppState? state =
context.findAncestorStateOfType<_MyAppState>();
    state?.setLocale(newLocale);
  }

  @override
  State<MyApp> createState() => _MyAppState();
}

class _MyAppState extends State<MyApp> {
  Locale? _locale;

  @override
  void initState() {
    super.initState();
    _loadSavedLocale();
  }

  Future<void> _loadSavedLocale() async {
    final prefs = await SharedPreferences.getInstance();
    final savedLangCode = prefs.getString('language_code');

    if (savedLangCode != null) {
      // Αν υπάρχει αποθηκευμένη γλώσσα, γινεται η χρηση της
      setState(() {
        _locale = Locale(savedLangCode);
      });
    } else {
      // Αν όχι, γινεται χρηση της γλώσσα του συστήματος
```

```
      final systemLangCode =
ui.PlatformDispatcher.instance.locale.languageCode;
      final isGreek = systemLangCode == 'el';

      setState(() {
        _locale = Locale(isGreek ? 'el' : 'en');
      });
    }
  }


  void setLocale(Locale newLocale) async {
    final prefs = await SharedPreferences.getInstance();
    await prefs.setString('language_code', newLocale.languageCode);
    setState(() {
      _locale = newLocale;
    });
  }

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Τεχνολογικό Μουσείο ΔΙΠΑΕ',
      debugShowCheckedModeBanner: false,
      locale: _locale,
      supportedLocales: const [
        Locale('el'),
        Locale('en'),
      ],
      localizationsDelegates: const [
        AppLocalizations.delegate,
        GlobalMaterialLocalizations.delegate,
        GlobalCupertinoLocalizations.delegate,
        GlobalWidgetsLocalizations.delegate,
      ],
      theme: ThemeData(
```

```
        primaryColor: const Color(0xFF005580),
        scaffoldBackgroundColor: const Color(0xFF224366),
        appBarTheme: const AppBarTheme(
          backgroundColor: Color(0xFF005580),
          titleTextStyle: TextStyle(
              color: Colors.white, fontSize: 20, fontWeight:
FontWeight.bold),
        ),
        floatingActionButtonTheme: const FloatingActionButtonThemeData(
          backgroundColor: Color(0xFFD41C1C),
        ),
      ),
      home: const SplashScreen(),
    );
  }
}

class SplashScreen extends StatefulWidget {
  const SplashScreen({Key? key}) : super(key: key);

  @override
  _SplashScreenState createState() => _SplashScreenState();
}

class _SplashScreenState extends State<SplashScreen> {
  @override
  void initState() {
    super.initState();
    WidgetsBinding.instance.addPostFrameCallback((_) {
      Future.delayed(const Duration(seconds: 3), () {
        if (!mounted) return;
        Navigator.pushReplacement(
          context,
          MaterialPageRoute(builder: (context) => const
ConnectionCheckScreen()),
        );
```

```
      });
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: const Color(0xFF163E66),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            Image.asset(
              'assets/playstore-icon.png',
              height: 100,
            ),
            const SizedBox(height: 20),
            Text(
              AppLocalizations.of(context)!.museumTitle,
              style: const TextStyle(fontSize: 20, color: Colors.white),
              textAlign: TextAlign.center,
            ),
            const SizedBox(height: 10),
            const CircularProgressIndicator(color: Colors.white),
          ],
        ),
      ),
    );
  }
}

class ConnectionCheckScreen extends StatefulWidget {
  const ConnectionCheckScreen({Key? key}) : super(key: key);

  @override
```

```
  _ConnectionCheckScreenState createState() =>
_ConnectionCheckScreenState();
}

class _ConnectionCheckScreenState extends State<ConnectionCheckScreen> {
  bool _isChecking = true;

  @override
  void initState() {
    super.initState();
    WidgetsBinding.instance.addPostFrameCallback((_) {
      _checkInternet();
    });
  }

  Future<void> _checkInternet() async {
    bool hasInternet = false;

    if (kIsWeb) {
      hasInternet = true;
    } else {
      hasInternet = await _hasRealInternet();
    }

    if (!mounted) return;

    if (hasInternet) {
      WidgetsBinding.instance.addPostFrameCallback((_) {
        Navigator.pushReplacement(
          context,
          MaterialPageRoute(builder: (_) => const MyHomePage()),
        );
      });
    } else {
      setState(() => _isChecking = false);
    }
```

```
  }

  Future<bool> _hasRealInternet() async {
    try {
      final result = await InternetAddress.lookup('google.com');
      return result.isNotEmpty && result[0].rawAddress.isNotEmpty;
    } catch (_) {
      return false;
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: const Color(0xFF224366),
      body: _isChecking
          ? const Center(child: CircularProgressIndicator(color:
Colors.white))
          : Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            const Icon(Icons.signal_wifi_connected_no_internet_4, size: 80,
color: Colors.red),
            const SizedBox(height: 20),
            Text(
              AppLocalizations.of(context)!.noInternet,
              style: const TextStyle(fontSize: 20, fontWeight:
FontWeight.bold, color: Colors.white),
            ),
            const SizedBox(height: 10),
            Text(AppLocalizations.of(context)!.noInternetMessage, style:
const TextStyle(color: Colors.white)),
            const SizedBox(height: 20),
            ElevatedButton(
              onPressed: _checkInternet,
```

```
              child: Text(AppLocalizations.of(context)!.retry, style: const
TextStyle(fontWeight: FontWeight.bold, color: Color(0xFF224366))),
            ),
          ],
        ),
      ),
    );
  }
}

class MyHomePage extends StatefulWidget {
  const MyHomePage({super.key});

  @override
  _MyHomePageState createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
  Map<String, dynamic>? randomExhibit;
  bool _showWebCategoryMenu = false;
  final FocusNode _searchFocusNode = FocusNode();
  bool _searchFocused = false;
  List<Map<String, dynamic>> searchResults = [];
  bool isSearching = false;
  TextEditingController searchController = TextEditingController();
  bool _isOffline = false;
  bool _isLoading = true;
  int _selectedIndex = 0;
  bool _showCategories = false;

  final List<Map<String, dynamic>> categories = [
    {
      'id': 'all',
      'name': 'Όλα τα εκθέματα',
      'name_en': 'All Exhibits',
      'icon': MdiIcons.formatListBulletedSquare,
```

```
  },
  {
    'id': 'computers',
    'name': 'Υπολογιστές',
    'name_en': 'Computers',
    'icon': MdiIcons.desktopClassic,
  },
  {
    'id': 'telecommunications',
    'name': 'Τηλεπικοινωνίες',
    'name_en': 'Telecommunications',
    'icon': MdiIcons.transmissionTower,
  },
  {
    'id': 'audio',
    'name': 'Ήχος & Μουσική',
    'name_en': 'Audio & Music',
    'icon': MdiIcons.music,
  },
  {
    'id': 'electronic_components',
    'name': 'Ηλεκτρονικά Εξαρτήματα',
    'name_en': 'Electronic Components',
    'icon': MdiIcons.resistor,
  },
];

@override
void initState() {
  _searchFocusNode.addListener(() {
    if (mounted) {
      final hasFocus = _searchFocusNode.hasFocus;

      Future.delayed(const Duration(milliseconds: 100), () {
        if (mounted) {
          setState(() {
```

```
            _searchFocused = hasFocus;
            _showWebCategoryMenu = kIsWeb && hasFocus;
          });
        }
      });

      if (hasFocus) {
        _searchExhibits('%');
      }
    }
  });
  super.initState();
  _fetchRandomExhibit().then((_) {
    if (mounted) {
      setState(() {
        _isLoading = false;
      });
    }
  });
  _startMonitoring();
  _checkAndShowHelpDialog();
}

Future<void> _checkAndShowHelpDialog() async {
  final prefs = await SharedPreferences.getInstance();
  final hasSeenHelp = prefs.getBool('hasSeenHelp') ?? false;
  if (!hasSeenHelp) {
    WidgetsBinding.instance.addPostFrameCallback((_) {
      _showHelpDialog(context);
    });
    await prefs.setBool('hasSeenHelp', true);
  }
}

Future<void> _fetchRandomExhibit() async {
  setState(() => _isLoading = true);
```

```
    try {
      final connectivityResult = await Connectivity().checkConnectivity();
      final hasInternet = connectivityResult != ConnectivityResult.none;

      if (!hasInternet) {
        ScaffoldMessenger.of(context).showSnackBar(
          SnackBar(
            content: Text(AppLocalizations.of(context)!.noInternet),
            backgroundColor: Colors.red,
          ),
        );
        return;
      }

      final response = await Supabase.instance.client
          .rpc('get_random_exhibit')
          .maybeSingle();

      if (response == null) return;

      final locale = Localizations.localeOf(context).languageCode;

      String name = response["name"] ?? "Άγνωστο Έκθεμα";
      String name_en = response["name_en"] ?? "Unknown Exhibit";
      String description = response["description"] ?? "Δεν υπάρχει
περιγραφή.";
      String description_en = response["description_en"] ?? "No
description.";
      String imageUrl = response["imageUrl"] ?? "No image.";
      setState(() {
        randomExhibit = {
          "id": response["id"],
          "name": name,
          "name_en": name_en,
          "description": description,
```

```
        "description_en": description_en,
        "imageUrl": imageUrl,
      };
    });
  } catch (e) {
    print("❌ Σφάλμα: $e");
  } finally {
    setState(() => _isLoading = false);
  }
}

Future<void> _fetchAllExhibitsByCategory(String categoryId) async {
  final locale = Localizations.localeOf(context).languageCode;

  final searchColumn = locale == 'en' ? 'name_en' : 'name';

  final response = await Supabase.instance.client
      .rpc('search_exhibits', params: {
    'search_term': '%',
    'lang': searchColumn,
    'category_id': categoryId,
  });

  final items = <Map<String, dynamic>>[];

  for (var exhibit in response) {
    items.add({
      "id": exhibit["id"] ?? "",
      "name": exhibit["name"] ?? "",
      "name_en": exhibit["name_en"] ?? "",
      "description": exhibit["description"] ?? "",
      "description_en": exhibit["description_en"] ?? "",
      "imageUrl": exhibit["imageUrl"] ?? "",
    });
  }
```

```
  if (mounted) {
    setState(() {
      searchResults = items;
    });
  }
}

Future<void> _fetchExhibitsByCategory(String categoryId) async {
  if (!mounted) return;
  setState(() => _isLoading = true);

  if (categoryId == 'all') {
    await _fetchRandomExhibit();
    if (!mounted) return;
    setState(() => _isLoading = false);
    return;
  }

  try {
    final response = await Supabase.instance.client
        .rpc('get_exhibits_by_category', params: {
      'category_input': categoryId,
    });

    if (!mounted || response.isEmpty) {
      setState(() => _isLoading = false);
      return;
    }

    final exhibit = response[Random().nextInt(response.length)];
    final locale = Localizations.localeOf(context).languageCode;

    setState(() {
      randomExhibit = {
        "id": exhibit["id"],
        "name": exhibit["name"],
```

```
        "name_en": exhibit["name_en"],
        "description": exhibit["description"],
        "description_en": exhibit["description_en"],
        "imageUrl": exhibit["imageUrl"],
        "category": exhibit["category"],
      };
    });
  } catch (e) {
    print("❌ Σφάλμα φόρτωσης κατηγορίας: $e");
  } finally {
    if (!mounted) return;
    setState(() => _isLoading = false);
  }
}

Timer? _debounce;

Future<void> _searchExhibits(String query) async {
  final trimmed = query.trim();
  final searchTerm = trimmed.isEmpty ? '%' : '%$trimmed';

  _debounce?.cancel();
  _debounce = Timer(const Duration(milliseconds: 300), () async {
    final locale = Localizations.localeOf(context).languageCode;
    String searchColumn = locale == 'en' ? 'name_en' : 'name';

    final categoryId = categories[_selectedIndex]['id'];
    final shouldFilterCategory = categoryId != 'all';

    final response = await Supabase.instance.client
        .rpc('search_exhibits', params: {
      'search_term': searchTerm,
      'lang': searchColumn,
      'category_id': shouldFilterCategory ? categoryId : 'all',
    });
```

```
      final translated = <Map<String, dynamic>>[];
      print("🔍 Response: $response");


      for (var exhibit in response) {
        translated.add({
          "id": exhibit["id"] ?? "",
          "name": exhibit["name"] ?? "",
          "name_en": exhibit["name_en"] ?? "",
          "description": exhibit["description"] ?? "Δεν υπάρχει
περιγραφή.",
          "description_en": exhibit["description_en"] ?? "No description.",
          "imageUrl": exhibit["imageUrl"] ?? "",
        });
        print("🔍 Response: ${exhibit['imageUrl']}");
      }

      if (mounted) {
        setState(() {
          searchResults = translated;
          isSearching = true;
        });
      }
    });
  }

  late StreamSubscription _subscription;

  void _startMonitoring() {
    _subscription =
Connectivity().onConnectivityChanged.listen((List<ConnectivityResult>
results) {
      final hasInternet = results.any((result) => result !=
ConnectivityResult.none);

      setState(() {
        _isOffline = !hasInternet;
```

```
    });

    if (_isOffline) {
      _showNoInternetSnackbar();
    }
  });
}

@override
void dispose() {
  _debounce?.cancel();
  _searchFocusNode.dispose();
  _subscription.cancel();
  super.dispose();
}

void _showNoInternetSnackbar() {
  ScaffoldMessenger.of(context).showSnackBar(
    SnackBar(
      content: Text(AppLocalizations.of(context)!.noInternet),
      duration: const Duration(seconds: 3),
      backgroundColor: Colors.red,
    ),
  );
}

void _showAboutDialog(BuildContext context) {
  showDialog(
    context: context,
    builder: (context) {
      return AlertDialog(
        backgroundColor: const Color(0xFF224366),
        title: Text(
          AppLocalizations.of(context)!.aboutAppTitle,
          style: const TextStyle(
            fontSize: 20,
```

```
              fontWeight: FontWeight.bold,
              color: Colors.white,
            ),
          ),
          content: Text(AppLocalizations.of(context)!.museumDes,
            style: const TextStyle(fontSize: 16, color: Colors.white),
          ),
          actions: [
            Row(
              mainAxisAlignment: MainAxisAlignment.spaceBetween,
              children: [
                TextButton.icon(
                  onPressed: () async {
                    final Uri url =
Uri.parse("https://github.com/Sotos2013/dipae-tech-museum");
                    if (await canLaunchUrl(url)) {
                      await launchUrl(url, mode:
LaunchMode.externalApplication);
                    } else {
                      ScaffoldMessenger.of(context).showSnackBar(
                        SnackBar(content:
Text(AppLocalizations.of(context)!.noGithub)),
                      );
                    }
                  },
                  icon: const FaIcon(FontAwesomeIcons.github, color:
Colors.white),
                  label: const Text(
                    "GitHub",
                    style: TextStyle(fontSize: 16, color: Colors.white),
                  ),
                  style: TextButton.styleFrom(
                    backgroundColor: Colors.black,
                    padding: const EdgeInsets.symmetric(horizontal: 15,
vertical: 10),
                    shape: RoundedRectangleBorder(
                      borderRadius: BorderRadius.circular(10),
```

```
                    ),
                  ),
                ),
                TextButton(
                  onPressed: () => Navigator.pop(context),
                  child: const Text(
                    "OK",
                    style: TextStyle(fontSize: 16, color: Colors.white),
                  ),
                ),
              ],
            ),
          ],
        );
      },
    );
  }

  void _showHelpDialog(BuildContext context) {
    showDialog(
      context: context,
      builder: (context) {
        return AlertDialog(
          backgroundColor: const Color(0xFF224366),
          title: Text(
            AppLocalizations.of(context)!.helpTitle,
            style: const TextStyle(fontSize: 20, fontWeight:
FontWeight.bold, color: Colors.white),
          ),
          content: Column(
            mainAxisSize: MainAxisSize.min,
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              _buildHelpRow(Icons.search,
AppLocalizations.of(context)!.searchHelp),
              _buildHelpRow(Icons.category,
AppLocalizations.of(context)!.categoryHelp),
```

```
              _buildHelpRow(Icons.info_outline,
AppLocalizations.of(context)!.tapExhibitHelp),
              _buildHelpRow(Icons.qr_code_scanner,
AppLocalizations.of(context)!.scanQrHelp),
              _buildHelpRow(Icons.language,
AppLocalizations.of(context)!.changeLanguageHelp),
              _buildHelpRow(Icons.touch_app_rounded,
AppLocalizations.of(context)!.tapHelp),
            ],
          ),
          actions: [
            TextButton(
              onPressed: () => Navigator.pop(context),
              child: const Text("OK", style: TextStyle(color:
Colors.white)),
            ),
          ],
        );
      },
    );
  }

  Widget _buildHelpRow(IconData icon, String text) {
    return Padding(
      padding: const EdgeInsets.symmetric(vertical: 6.0),
      child: Row(
        children: [
          Icon(icon, color: Colors.white),
          const SizedBox(width: 10),
          Expanded(
            child: Text(
              text,
              style: const TextStyle(color: Colors.white),
            ),
          ),
        ],
      ),
```

```
  );
}

Widget _buildCategoriesMenu(BuildContext context) {
  final locale = Localizations.localeOf(context).languageCode;

  return AnimatedContainer(
    duration: const Duration(milliseconds: 300),
    height: _showCategories ? 200 : 0,
    decoration: BoxDecoration(
      color: const Color(0xFF163E66),
      borderRadius: BorderRadius.circular(10),
      boxShadow: [
        BoxShadow(
          color: Colors.black.withOpacity(0.2),
          blurRadius: 10,
          spreadRadius: 2,
        ),
      ],
    ),
    child: SingleChildScrollView(
      child: Column(
        children: categories.map((category) => ListTile(
          leading: Icon(category['icon'], color: Colors.white),
          title: Text(
            locale == 'en' ? category['name_en'] : category['name'],
            style: const TextStyle(color: Colors.white),
          ),
          onTap: () {
            setState(() {
              _selectedIndex = categories.indexOf(category);
              _showCategories = false;
            });
            _fetchExhibitsByCategory(category['id']);
          },
        )).toList(),
```

```
      ),
    ),
  );
}

@override
Widget build(BuildContext context) {
  return PopScope(
    canPop: false,
    onPopInvoked: (didPop) async {
      // Αν υπάρχει αναζήτηση ή οποιαδήποτε κατηγορία έχει επιλεγεί
      if (isSearching || _selectedIndex != 0) {
        setState(() {
          isSearching = false;
          searchController.clear();
          _selectedIndex = 0;
        });
        await _fetchRandomExhibit(); // Επαναφορά αρχικής οθόνης
      } else {
        // Αν είμαστε ήδη στην αρχική, τότε επιτρέπουμε έξοδο
        Navigator.of(context).maybePop();
      }
    },
    child: Scaffold(
      appBar: AppBar(
        title: GestureDetector(
          onTap: () {
            Navigator.pushReplacement(
              context,
              MaterialPageRoute(builder: (_) => const MyHomePage()),
            );
          },
          child: Text(
            AppLocalizations.of(context)!.museumTitle,
            style: const TextStyle(
              color: Colors.white,
```

```
                fontWeight: FontWeight.bold,
              ),
            ),
          ),
          actions: [
            IconButton(
              icon: const Icon(Icons.help_outline, color: Colors.white),
              onPressed: () => _showHelpDialog(context),
            ),
            IconButton(
              icon: const Icon(Icons.info_outline, color: Colors.white),
              onPressed: () => _showAboutDialog(context),
            ),
            IconButton(
              icon: const Icon(Icons.language, color: Colors.white),
              onPressed: () async {
                final currentLang =
Localizations.localeOf(context).languageCode;
                final newLocale = currentLang == 'el' ? const Locale('en')
: const Locale('el');
                MyApp.setLocale(context, newLocale);
              },
            ),
          ],
        ),
        body: _isLoading
            ? const Center(child: CircularProgressIndicator(color:
Colors.white))
            : RefreshIndicator(
          onRefresh: _onRefresh,
          color: const Color(0xFFD41C1C),
          child: GestureDetector(
            onTap: () {
              FocusScope.of(context).unfocus();
              setState(() => _showCategories = false);
            },
            child: Stack(
```

```
              children: [
                ListView(
                  padding: const EdgeInsets.all(20.0),
                  children: [
                    _buildSearchBar(context),
                    const SizedBox(height: 10),
                    _buildCategoriesMenu(context),
                    const SizedBox(height: 20),
                    if (isSearching)
                      _buildSearchResults(context)
                    else if (randomExhibit != null)
                      _buildMainInfo(context),
                  ],
                ),
              ],
            ),
          ),
        ),
        floatingActionButton: FloatingActionButton(
          onPressed: () {
            Navigator.push(
              context,
              MaterialPageRoute(builder: (context) => const
QRScannerScreen()),
            );
          },
          child: const Icon(Icons.qr_code_scanner, color: Colors.white),
        ),
      ),
    );
  }

  Widget _buildMainInfo(BuildContext context) {
    return Column(
      children: [
        if (randomExhibit != null) _buildRandomExhibitCard(context),
```

```
        const SizedBox(height: 20),
        _buildMuseumInfoSection(context),
        const SizedBox(height: 20),
        _buildUniversityLogo(context),
        const SizedBox(height: 20),
        _buildFinalQuizButton(context),
        const SizedBox(height: 20),
        _buildFeedbackButton(context),
      ],
    );
  }

  Widget _buildSearchBar(BuildContext context) {
    final locale = Localizations.localeOf(context).languageCode;

    return Padding(
      padding: const EdgeInsets.symmetric(horizontal: 16, vertical: 8),
      child: Row(
        children: [
          Expanded(
            child: TextField(
              focusNode: _searchFocusNode,
              controller: searchController,
              decoration: InputDecoration(
                hintText: AppLocalizations.of(context)!.searchPlaceholder,
                filled: true,
                fillColor: Colors.white,
                prefixIcon: const Icon(Icons.search, color:
Color(0xFF005580)),
                border: OutlineInputBorder(
                  borderRadius: BorderRadius.circular(20),
                  borderSide: BorderSide.none,
                ),
                contentPadding: const EdgeInsets.symmetric(vertical: 14,
horizontal: 16),
              ),
```

```
              onChanged: _searchExhibits,
            ),
          ),
          IconButton(
            icon: Icon(categories[_selectedIndex]['icon'], color:
Colors.white),
            tooltip: AppLocalizations.of(context)!.chooseCategory,
            onPressed: () async {
              // Καθυστέρηση ώστε να μη χαθεί το focus και εξαφανιστεί το
menu αμέσως
              await Future.delayed(const Duration(milliseconds: 100));

              final selected = await showMenu<String>(
                context: context,
                position: const RelativeRect.fromLTRB(100, 100, 0, 0),
                items: categories.map((category) {
                  final title = locale == 'en' ? category['name_en'] :
category['name'];
                  return PopupMenuItem<String>(
                    value: category['id'],
                    child: Row(
                      children: [
                        Icon(category['icon'], color: Colors.black),
                        const SizedBox(width: 10),
                        Text(title),
                      ],
                    ),
                  );
                }).toList(),
              );

              if (!mounted || selected == null) return;

              final index = categories.indexWhere((c) => c['id'] ==
selected);
              setState(() => _selectedIndex = index);
```

```
              final query = searchController.text.trim();
              if (query.isEmpty) {
                await _fetchAllExhibitsByCategory(selected);
                setState(() => isSearching = true);
              } else {
                _searchExhibits(query);
              }
            },
          ),
        ],
      ),
    );
  }

  Widget _buildSearchResults(BuildContext context) {
    if (searchResults.isEmpty) {
      return Center(
        child: Text(
          AppLocalizations.of(context)!.noResults,
          style: const TextStyle(color: Colors.white, fontSize: 16),
        ),
      );
    }
    return Column(
      children: searchResults.map((exhibit) => _buildExhibitTile(context,
exhibit)).toList(),
    );
  }

  Widget _buildExhibitTile(BuildContext context, Map<String, dynamic>
exhibit) {
    final locale = Localizations.localeOf(context).languageCode;
    final displayName = locale == 'en' ? exhibit['name_en'] :
exhibit['name'];
    final displayDescription = locale == 'en' ? exhibit['description_en'] :
exhibit['description'];
```

```
    return Card(
      margin: const EdgeInsets.symmetric(vertical: 8, horizontal: 16),
      color: const Color(0xFF163E66),
      child: ListTile(
        onTap: () {
          Navigator.push(
            context,
            MaterialPageRoute(
              builder: (_) => QRInfoScreen(
                id: exhibit['id'] ?? '',
                name: exhibit['name'] ?? '',
                name_en: exhibit['name_en'] ?? '',
                description: exhibit['description'] ?? '',
                description_en: exhibit['description_en'] ?? '',
                imageUrl: exhibit['imageUrl'] ?? '',
              ),
            ),
          );
        },
        contentPadding: const EdgeInsets.symmetric(horizontal: 16,
vertical: 8),
        leading: ClipRRect(
          borderRadius: BorderRadius.circular(8),
          child: Image.network(
            exhibit['imageUrl'] ?? '',
            width: 50,
            height: 50,
            fit: BoxFit.cover,
            errorBuilder: (_, __, ___) => const Icon(
              Icons.broken_image,
              size: 50,
              color: Color(0xFFD41C1C),
            ),
          ),
        ),
        title: Text(
```

```
        displayName,
        style: const TextStyle(
          fontSize: 16,
          fontWeight: FontWeight.bold,
          color: Colors.white,
        ),
        maxLines: 2,
        overflow: TextOverflow.ellipsis,
      ),
      subtitle: Text(
        displayDescription,
        style: const TextStyle(fontSize: 14, color: Colors.white70),
        maxLines: 2,
        overflow: TextOverflow.ellipsis,
      ),
    ),
  );
}

Future<void> _onRefresh() async {
  final connectivityResult = await Connectivity().checkConnectivity();
  final hasInternet = connectivityResult != ConnectivityResult.none;

  if (hasInternet) {
    await _fetchRandomExhibit();
  } else {
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(
        content: Text(AppLocalizations.of(context)!.noInternet),
        backgroundColor: const Color(0xFFD41C1C),
        duration: const Duration(seconds: 3),
      ),
    );
  }
}
```

```
  Widget _buildFeedbackButton(BuildContext context) {
    return ElevatedButton(
      onPressed: () async {
        final locale = Localizations.localeOf(context).languageCode;

        final url = Uri.parse(
          locale == 'en'
              ?
"https://docs.google.com/forms/d/e/1FAIpQLScNlIzKkxiP1jgLTaUS-
3xiQjYt6O7UMpZ3rm1gGygJazSiKg/viewform"
              : "https://docs.google.com/forms/d/e/1FAIpQLSeve-
CdFpu5gper6D2QnmHu6cs99fqvGeK7A2UCNmk6JRZWjQ/viewform",
        );

        if (!await launchUrl(url, mode: LaunchMode.externalApplication)) {
          ScaffoldMessenger.of(context).showSnackBar(
            SnackBar(content:
Text(AppLocalizations.of(context)!.noInternet)),
          );
        }
      },
      style: ElevatedButton.styleFrom(
        backgroundColor: const Color(0xFFD41C1C),
        foregroundColor: Colors.white,
        padding: const EdgeInsets.symmetric(horizontal: 20, vertical: 15),
        textStyle: const TextStyle(fontSize: 16, fontWeight:
FontWeight.bold),
        shape: RoundedRectangleBorder(borderRadius:
BorderRadius.circular(10)),
      ),
      child: Text(AppLocalizations.of(context)!.questionnaire),
    );
  }

  Widget _buildFinalQuizButton(BuildContext context) {
    return ElevatedButton.icon(
      onPressed: () {
```

```
        Navigator.push(context, MaterialPageRoute(builder: (_) => const
FinalQuizScreen()));
      },
      icon: const Icon(Icons.school),
      label: Text(AppLocalizations.of(context)!.finalQuiz),
      style: ElevatedButton.styleFrom(
        backgroundColor: Colors.green,
        foregroundColor: Colors.white,
        padding: const EdgeInsets.symmetric(horizontal: 20, vertical: 15),
        textStyle: const TextStyle(fontSize: 16, fontWeight:
FontWeight.bold),
      ),
    );
  }

  Widget _buildUniversityLogo(BuildContext context) {
    return Column(
      children: [
        Image.asset('assets/ihu_logo.png', height: 80),
        Text(
          AppLocalizations.of(context)!.ihuName,
          style: const TextStyle(fontSize: 16, color: Colors.white),
        ),
      ],
    );
  }

  Widget _buildMuseumInfoSection(BuildContext context) {
    return IgnorePointer(
      child: Container(
        decoration: BoxDecoration(
          color: const Color(0xFF005580),
          borderRadius: BorderRadius.circular(15),
        ),
        padding: const EdgeInsets.all(15),
        child: Column(
```

```
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            Text(
              AppLocalizations.of(context)!.museumInfoTitle,
              style: const TextStyle(fontSize: 18, fontWeight:
FontWeight.bold, color: Colors.white),
            ),
            const SizedBox(height: 10),
            Text(
              AppLocalizations.of(context)!.museumInfo1,
              style: const TextStyle(fontSize: 16, color: Colors.white),
            ),
            const SizedBox(height: 15),
            Text(
              AppLocalizations.of(context)!.trainStoryTitle,
              style: const TextStyle(fontSize: 18, fontWeight:
FontWeight.bold, color: Colors.white),
            ),
            const SizedBox(height: 10),
            Text(
              AppLocalizations.of(context)!.trainStory,
              style: const TextStyle(fontSize: 16, color: Colors.white),
            ),
            const SizedBox(height: 15),
            Text(
              AppLocalizations.of(context)!.discoverTitle,
              style: const TextStyle(fontSize: 18, fontWeight:
FontWeight.bold, color: Colors.white),
            ),
            const SizedBox(height: 10),
            Text(
              AppLocalizations.of(context)!.discoverItems,
              style: const TextStyle(fontSize: 16, color: Colors.white),
            ),
          ],
        ),
      ),
```

```
    );
  }

  Widget _buildRandomExhibitCard(BuildContext context) {
    return GestureDetector(
      onTap: () {
        Navigator.push(
          context,
          MaterialPageRoute(
            builder: (_) => QRInfoScreen(
              id: randomExhibit!['id'] ?? '',
              name: randomExhibit!['name'] ?? '',
              name_en: randomExhibit!['name_en'] ?? '',
              description: randomExhibit!['description'] ?? '',
              description_en: randomExhibit!['description_en'] ?? '',
              imageUrl: randomExhibit!['imageUrl'] ?? '',
            ),
          ),
        ).then((_) => _fetchRandomExhibit());
      },
      child: Card(
        elevation: 5,
        shape: RoundedRectangleBorder(borderRadius:
BorderRadius.circular(15)),
        child: Column(
          children: [
            ClipRRect(
              borderRadius: const BorderRadius.vertical(top:
Radius.circular(15)),
              child: randomExhibit!['imageUrl'] != null &&
randomExhibit!['imageUrl'].toString().isNotEmpty
                  ? Image.network(
                randomExhibit!['imageUrl'],
                height: 150,
                width: double.infinity,
                fit: BoxFit.cover,
```

```
                errorBuilder: (_, __, ___) => const
Icon(Icons.broken_image, size: 100, color: Colors.red),
              )
                  : const Icon(Icons.broken_image, size: 150, color:
Colors.red),
            ),
            Padding(
              padding: const EdgeInsets.all(10.0),
              child: Text(
                "${AppLocalizations.of(context)!.randomExhibit} :
${(Localizations.localeOf(context).languageCode == 'en'
                    ? randomExhibit!['name_en']
                    : randomExhibit!['name']) ?? ''}",
                style: const TextStyle(fontSize: 18, fontWeight:
FontWeight.bold),
              ),
            ),
          ],
        ),
      ),
    );
  }
}
```

Final_quiz_screen.dart

```
import 'package:flutter/material.dart';
import 'package:supabase_flutter/supabase_flutter.dart';
import 'package:untitled1/translation_helper.dart';
import 'package:flutter_gen/gen_l10n/app_localizations.dart';

class FinalQuizScreen extends StatefulWidget {
  const FinalQuizScreen({Key? key}) : super(key: key);

  @override
  State<FinalQuizScreen> createState() => _FinalQuizScreenState();
}
```

```
class _FinalQuizScreenState extends State<FinalQuizScreen> {
  List<Map<String, dynamic>> questions = [];
  int currentQuestionIndex = 0;
  int score = 0;
  bool isLoading = true;

  @override
  void didChangeDependencies() {
    super.didChangeDependencies();
    _fetchRandomQuestions(Localizations.localeOf(context).languageCode);
  }

  Future<void> _fetchRandomQuestions(String locale) async {
    try {
      final response = await Supabase.instance.client
          .rpc('get_random_quiz_questions', params: {'limit_count': 5});

      final translated = await Future.wait(response.map<Future<Map<String,
dynamic>>>((question) async {
        final originalQ = question['question'];
        final answers = List<Map<String,
dynamic>>.from(question['answers']);

        String translatedQ = originalQ;

        if (locale == 'en') {
          translatedQ = await TranslationHelper.translate(originalQ, 'el',
'en');
          for (var answer in answers) {
            answer['text'] = await
TranslationHelper.translate(answer['text'], 'el', 'en');
          }
        }
        answers.shuffle();
        return {
          'question': translatedQ,
          'answers': answers,
```

```
        };
      }));

      setState(() {
        questions = List<Map<String, dynamic>>.from(translated);
        isLoading = false;
      });
    } catch (e) {
      print("❌ Σφάλμα στο final quiz: $e");
      setState(() => isLoading = false);
    }
  }

  void _checkAnswer(bool isCorrect) {
    if (isCorrect) score++;

    if (currentQuestionIndex < questions.length - 1) {
      setState(() => currentQuestionIndex++);
    } else {
      _showResults();
    }
  }

  void _showResults() {
    showDialog(
      context: context,
      builder: (_) => AlertDialog(
        title: Text(AppLocalizations.of(context)!.quizComplete),
        content: Text("$score / ${questions.length}"),
        actions: [
          TextButton(
            onPressed: () => Navigator.popUntil(context, (route) =>
route.isFirst),
            child: const Text("OK"),
          ),
        ],
```

```
      ),
    );
  }

  @override
  Widget build(BuildContext context) {
    if (isLoading) {
      return const Scaffold(
        body: Center(child: CircularProgressIndicator(color:
Colors.white,)),
      );
    }

    if (questions.isEmpty) {
      return Scaffold(
        appBar: AppBar(
          title: Text(AppLocalizations.of(context)!.finalQuiz),
        ),
        body: Center(
          child: Text(
            AppLocalizations.of(context)!.noQuestions,
            style: const TextStyle(fontSize: 18, fontWeight:
FontWeight.bold),
          ),
        ),
      );
    }

    final q = questions[currentQuestionIndex];
    return Scaffold(
      appBar: AppBar(
        backgroundColor: Color(0xFF005580),
        title: Text("${AppLocalizations.of(context)!.question}
${currentQuestionIndex + 1} / ${questions.length}"),
      ),
      body: Padding(
        padding: const EdgeInsets.all(20.0),
```

```
        child: Center(
          child: Card(
            elevation: 6,
            shape: RoundedRectangleBorder(borderRadius:
BorderRadius.circular(15)),
            child: Padding(
              padding: const EdgeInsets.all(20.0),
              child: Column(
                mainAxisSize: MainAxisSize.min,
                crossAxisAlignment: CrossAxisAlignment.stretch,
                children: [
                  Text(
                    q['question'],
                    textAlign: TextAlign.center,
                    style: const TextStyle(
                      fontSize: 20,
                      fontWeight: FontWeight.bold,
                      color: Color(0xFF224366),
                    ),
                  ),
                  const SizedBox(height: 20),
                  ...(q['answers'] as List).map((a) => Padding(
                    padding: const EdgeInsets.symmetric(vertical: 6.0),
                    child: Center(
                      child: ElevatedButton(
                        onPressed: () => _checkAnswer(a['correct']),
                        style: ElevatedButton.styleFrom(
                          backgroundColor: const Color(0xFF005580),
                          foregroundColor: Colors.white,
                          minimumSize: const Size(double.infinity, 40),
                          textStyle: const TextStyle(fontSize: 15),
                          shape: RoundedRectangleBorder(
                            borderRadius: BorderRadius.circular(30),
                          ),
                        ),
                        child: Text(
```

```
                          a['text'],
                          textAlign: TextAlign.center,
                        ),
                      ),
                    ),
                  )),
                ],
              ),
            ),
          ),
        ),
      ),
    );
  }
}
```

Qr_info_screen.dart

```
import 'package:flutter/material.dart';
import 'package:untitled1/quiz_screen.dart';
import 'package:flutter_gen/gen_l10n/app_localizations.dart';
import 'main.dart';

class QRInfoScreen extends StatefulWidget {
  final String id;
  final String name;
  final String name_en;
  final String description;
  final String description_en;
  final String imageUrl;

  const QRInfoScreen({
    Key? key,
    required this.id,
    required this.name,
```

```
    required this.name_en,
    required this.description,
    required this.description_en,
    required this.imageUrl,
  }) : super(key: key);

  @override
  State<QRInfoScreen> createState() => _QRInfoScreenState();
}

class _QRInfoScreenState extends State<QRInfoScreen> {
  @override
  Widget build(BuildContext context) {
    final locale = Localizations.localeOf(context).languageCode;
    final encodedUrl = Uri.encodeFull(widget.imageUrl);

    final name = locale == 'en'
        ? (widget.name_en.isNotEmpty ? widget.name_en : widget.name)
        : (widget.name.isNotEmpty ? widget.name : widget.name_en);

    final description = locale == 'en'
        ? (widget.description_en.isNotEmpty ? widget.description_en :
widget.description)
        : (widget.description.isNotEmpty ? widget.description :
widget.description_en);

    return Scaffold(
      appBar: AppBar(
        title: Text(
          AppLocalizations.of(context)!.exhibitInfoTitle,
          style: const TextStyle(color: Colors.white),
        ),
        actions: [
          IconButton(
            icon: const Icon(Icons.language, color: Colors.white),
            onPressed: () async {
```

```
              final newLocale = locale == 'el' ? const Locale('en') : const
Locale('el');
              MyApp.setLocale(context, newLocale);
              setState(() {}); // κάνει rebuild για να αλλάξει η περιγραφή
            },
          ),
        ],
        backgroundColor: const Color(0xFF005580),
      ),
      backgroundColor: const Color(0xFF224366),
      body: Center(
        child: SingleChildScrollView(
          padding: const EdgeInsets.symmetric(horizontal: 20.0),
          child: Card(
            elevation: 8,
            shape: RoundedRectangleBorder(
              borderRadius: BorderRadius.circular(15),
            ),
            child: Padding(
              padding: const EdgeInsets.all(20.0),
              child: Column(
                mainAxisSize: MainAxisSize.min,
                crossAxisAlignment: CrossAxisAlignment.center,
                children: [
                  ClipRRect(
                    borderRadius: BorderRadius.circular(10),
                    child: Image.network(
                      encodedUrl,
                      height: 200,
                      fit: BoxFit.cover,
                      errorBuilder: (context, error, stackTrace) {
                        return Column(
                          children: [
                            const Icon(Icons.broken_image, size: 100,
color: Colors.red),
                            const SizedBox(height: 10),
```

```
                        Text(
                          AppLocalizations.of(context)!.imageLoadError,
                          style: const TextStyle(color: Colors.red),
                        ),
                      ],
                    );
                  },
                ),
              ),
              const SizedBox(height: 15),
              Text(
                name,
                textAlign: TextAlign.center,
                style: const TextStyle(
                  fontSize: 22,
                  fontWeight: FontWeight.bold,
                  color: Color(0xFF224366),
                ),
              ),
              const SizedBox(height: 10),
              Text(
                description,
                textAlign: TextAlign.center,
                style: const TextStyle(fontSize: 18, color:
Colors.black87),
              ),
              const SizedBox(height: 20),
              ElevatedButton.icon(
                onPressed: () => Navigator.pop(context),
                icon: const Icon(Icons.arrow_back, color:
Colors.white),
                label: Text(
                  AppLocalizations.of(context)!.backButton,
                  style: const TextStyle(color: Colors.white),
                ),
                style: ElevatedButton.styleFrom(
```

```
                    backgroundColor: const Color(0xFFD41C1C),
                    padding: const EdgeInsets.symmetric(horizontal: 20,
vertical: 12),
                    shape: RoundedRectangleBorder(
                      borderRadius: BorderRadius.circular(10),
                    ),
                  ),
                ),
                const SizedBox(height: 10),
                ElevatedButton.icon(
                  onPressed: () {
                    Navigator.push(
                      context,
                      MaterialPageRoute(
                        builder: (context) => QuizScreen(qrCode:
widget.id),
                      ),
                    );
                  },
                  icon: const Icon(Icons.quiz, color: Colors.white),
                  label: Text(
                    AppLocalizations.of(context)!.startQuizButton,
                    style: const TextStyle(color: Colors.white),
                  ),
                  style: ElevatedButton.styleFrom(
                    backgroundColor: Colors.green,
                    padding: const EdgeInsets.symmetric(horizontal: 20,
vertical: 12),
                    shape: RoundedRectangleBorder(
                      borderRadius: BorderRadius.circular(10),
                    ),
                  ),
                ),
              ],
            ),
          ),
        ),
```

```
        ),
      ),
    );
  }
}
```

Qr_scanner_screen.dart

```
import 'package:flutter/foundation.dart';
import 'package:flutter/material.dart';
import 'package:mobile_scanner/mobile_scanner.dart';
import 'package:connectivity_plus/connectivity_plus.dart';
import 'package:supabase_flutter/supabase_flutter.dart';
import 'dart:async';
import 'package:flutter_gen/gen_l10n/app_localizations.dart';
import 'qr_info_screen.dart';

class QRScannerScreen extends StatefulWidget {
  const QRScannerScreen({Key? key}) : super(key: key);

  @override
  State<QRScannerScreen> createState() => _QRScannerScreenState();
}

class _QRScannerScreenState extends State<QRScannerScreen> {
  final MobileScannerController cameraController = MobileScannerController(
    facing: CameraFacing.back,
  );

  bool _isScanning = true;
  bool _hasShownNoInternetMessage = false;
  bool _isFlashOn = false;
  bool _hasShownInvalidQrMessage = false;
  Timer? _debounceTimer;
```

```
  @override
  void initState() {
    super.initState();
  }

  @override
  void dispose() {
    cameraController.dispose();
    _debounceTimer?.cancel();
    super.dispose();
  }

  Future<bool> _checkInternetConnection() async {
    var connectivityResult = await Connectivity().checkConnectivity();
    return connectivityResult != ConnectivityResult.none;
  }

  Future<void> _checkQRCode(String code) async {
    bool hasInternet = await _checkInternetConnection();

    if (!hasInternet) {
      if (!_hasShownNoInternetMessage) {
        _hasShownNoInternetMessage = true;
        ScaffoldMessenger.of(context).showSnackBar(
          SnackBar(
            content: Text(
              AppLocalizations.of(context)!.noInternetMessage,
              style: const TextStyle(color: Colors.white, fontWeight:
FontWeight.bold),
            ),
            backgroundColor: Colors.red,
            duration: const Duration(seconds: 3),
          ),
        );
        Future.delayed(const Duration(seconds: 3), () {
          _hasShownNoInternetMessage = false;
```

```
        });
      }
      return;
    }

    try {
      final List<dynamic> result = await Supabase.instance.client
          .rpc('get_qr_code_info', params: {'qr_id': code});
      final response = result.isNotEmpty ? result.first : null;

      if (response != null) {
        Navigator.pushReplacement(
          context,
          MaterialPageRoute(
            builder: (context) => QRInfoScreen(
              id: response['id'] ?? '',
              name: response['name'] ?? '',
              name_en: response['name_en'] ?? '',
              description: response['description'] ?? '',
              description_en: response['description_en'] ?? '',
              imageUrl: response['imageUrl'] ?? '',
            ),
          ),
        );
      } else {
        if (!_hasShownInvalidQrMessage) {
          _hasShownInvalidQrMessage = true;
          ScaffoldMessenger.of(context).showSnackBar(
            SnackBar(
              content: Text(
                AppLocalizations.of(context)!.invalidQrMessage,
                style: const TextStyle(color: Colors.white, fontWeight:
FontWeight.bold),
              ),
              backgroundColor: Colors.red,
              duration: const Duration(seconds: 3),
```

```
          ),
        );
        Future.delayed(const Duration(seconds: 3), () {
          _hasShownInvalidQrMessage = false;
        });
      }
    }
  } catch (e) {
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(
        content: Text(AppLocalizations.of(context)!.qrSearchError),
        backgroundColor: Colors.red,
      ),
    );
  }
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      backgroundColor: const Color(0xFF224366),
      title: Text(
        AppLocalizations.of(context)!.qrScannerTitle,
        style: const TextStyle(color: Colors.white),
      ),
      actions: [
        if (!kIsWeb)
          IconButton(
            icon: Icon(
              _isFlashOn ? Icons.flash_on : Icons.flash_off,
              color: Colors.white,
            ),
            onPressed: () {
              setState(() {
                _isFlashOn = !_isFlashOn;
```

```
                cameraController.toggleTorch();
              });
            },
          ),
        ],
      ),
      body: Stack(
        alignment: Alignment.center,
        children: [
          MobileScanner(
            controller: cameraController,
            onDetect: (capture) {
              if (!_isScanning) return;
              if (_debounceTimer?.isActive ?? false) return;

              _debounceTimer = Timer(const Duration(milliseconds: 800), ()
{
                final String? code = capture.barcodes.first.rawValue;
                if (code != null) {
                  setState(() => _isScanning = false);

                  _checkQRCode(code).then((_) {
                    setState(() => _isScanning = true);
                  });
                }
              });
            },
          ),
          Positioned(
            top: MediaQuery.of(context).size.height * 0.3,
            child: AnimatedContainer(
              duration: const Duration(milliseconds: 500),
              width: 250,
              height: 250,
              decoration: BoxDecoration(
                border: Border.all(color: Colors.white, width: 4),
```

```
                borderRadius: BorderRadius.circular(20),
                boxShadow: [
                  BoxShadow(
                    color: Colors.white.withOpacity(0.5),
                    blurRadius: 10,
                    spreadRadius: 1,
                  ),
                ],
              ),
            ),
          ),
        ],
      ),
    );
  }
}
```

Quiz_screen.dart

```
import 'package:flutter/material.dart';
import 'package:supabase_flutter/supabase_flutter.dart';
import 'package:untitled1/translation_helper.dart';
import 'package:flutter_gen/gen_l10n/app_localizations.dart';

class QuizScreen extends StatefulWidget {
  final String qrCode;

  const QuizScreen({Key? key, required this.qrCode}) : super(key: key);

  @override
  _QuizScreenState createState() => _QuizScreenState();
}

class _QuizScreenState extends State<QuizScreen> {
```

```
  List<Map<String, dynamic>> questions = [];
  int currentQuestionIndex = 0;
  int score = 0;
  bool isLoading = true;

  @override
  void initState() {
    super.initState();
  }
  String? _currentLocale;

  @override
  void didChangeDependencies() {
    super.didChangeDependencies();

    final locale = Localizations.localeOf(context).languageCode;

    if (_currentLocale != locale) {
      _currentLocale = locale;
      _fetchQuestions(locale);
    }
  }

  Future<void> _fetchQuestions(String locale) async {
    try {
      final List<dynamic> response = await Supabase.instance.client
          .rpc('get_quiz_questions', params: {'qr_id': widget.qrCode});

      if (response.isNotEmpty) {
        final translated = await Future.wait(response.map((question) async
{
          final originalQ = question['question'];
          final answers = List<Map<String,
dynamic>>.from(question['answers']);
          String translatedQ = originalQ;
```

```
          if (locale == 'en') {
            translatedQ = await TranslationHelper.translate(originalQ,
'el', 'en');
            for (var answer in answers) {
              answer['text'] = await
TranslationHelper.translate(answer['text'], 'el', 'en');
            }
          }
          answers.shuffle();
          return {
            'question': translatedQ,
            'answers': answers,
          };
        }));

        setState(() {
          questions = List<Map<String, dynamic>>.from(translated);
          isLoading = false;
        });
      } else {
        setState(() => isLoading = false);
      }
    } catch (e) {
      print("❌ Σφάλμα φόρτωσης ερωτήσεων: $e");
      setState(() => isLoading = false);
    }
  }

  void _checkAnswer(bool isCorrect) {
    if (isCorrect) {
      score++;
    }

    // Ελέγχουμε αν υπάρχει άλλη ερώτηση ή αν τελείωσε το Quiz
    if (currentQuestionIndex < questions.length - 1) {
      setState(() {
```

```
      currentQuestionIndex++;
    });
  } else {
    _showResults();
  }
}

void _showResults() {
  showDialog(
    context: context,
    builder: (context) {
      return AlertDialog(
        title: Text(AppLocalizations.of(context)!.quizComplete),
        content: Text("$score / ${questions.length}"),
        actions: [
          TextButton(
            onPressed: () {
              Navigator.pop(context);
              Navigator.pop(context); // Επιστροφή στην προηγούμενη οθόνη
            },
            child: const Text("OK",
              style: const TextStyle(
                fontSize: 15,
                fontWeight: FontWeight.bold,
                color: Color(0xFF224366),
              ),
            ),
          ),
        ],
      );
    },
  );
}

@override
Widget build(BuildContext context) {
```

```
    if (isLoading) {
      return Scaffold(
        appBar: AppBar(title: const Text("Quiz")),
        body: const Center(child: CircularProgressIndicator(color:
Colors.white,)),
      );
    }

    if (questions.isEmpty) {
      return Scaffold(
        appBar: AppBar(title: const Text("Quiz")),
        body: Center(
          child: Text(
            AppLocalizations.of(context)!.noQuestions,
            style: const TextStyle(fontSize: 18, fontWeight:
FontWeight.bold),
          ),
        ),
      );
    }

    var question = questions[currentQuestionIndex];

    return Scaffold(
      appBar: AppBar(
        title: Text("${AppLocalizations.of(context)!.question}
${currentQuestionIndex + 1} / ${questions.length}"),
        backgroundColor: Color(0xFF005580),
      ),
      body: Padding(
        padding: const EdgeInsets.all(20.0),
        child: Center(
          child: Card(
            elevation: 6,
            shape: RoundedRectangleBorder(borderRadius:
BorderRadius.circular(15)),
            child: Padding(
```

```
padding: const EdgeInsets.all(20.0),
child: Column(
  mainAxisSize: MainAxisSize.min,
  crossAxisAlignment: CrossAxisAlignment.stretch,
  children: [
    Text(
      question['question'],
      textAlign: TextAlign.center,
      style: const TextStyle(
        fontSize: 20,
        fontWeight: FontWeight.bold,
        color: Color(0xFF224366),
      ),
    ),
    const SizedBox(height: 20),
    ...(question['answers'] as List<dynamic>).map((answer) {
      return Padding(
        padding: const EdgeInsets.symmetric(vertical: 6.0),
        child: Center(
          child: ElevatedButton(
            onPressed: () => _checkAnswer(answer['correct']),
            style: ElevatedButton.styleFrom(
              backgroundColor: const Color(0xFF005580),
              foregroundColor: Colors.white,
              minimumSize: const Size(double.infinity, 40),
              textStyle: const TextStyle(fontSize: 15),
              shape: RoundedRectangleBorder(
                borderRadius: BorderRadius.circular(30),
              ),
            ),
            child: Center(
              child: Text(
                answer['text'],
                textAlign: TextAlign.center,
              ),
            ),
```

```
                        ),
                      ),
                    );
                  }),
                ],
              ),
            ),
          ),
        ),
      ),
    );
  }
}
```

Translation_helper.dart

```
import 'dart:convert';
import 'package:http/http.dart' as http;
import 'package:shared_preferences/shared_preferences.dart';
import 'package:flutter_dotenv/flutter_dotenv.dart';

class TranslationHelper {
  static const String _email = 'www.sotihatzi@gmail.com';

  static Future<String> translate(String text, String from, String to)
async {
    final prefs = await SharedPreferences.getInstance();
    final cacheKey = 'translation_${from}_$to:$text';
    final _apiKey = dotenv.env['MYMEMORY_API_KEY'];
    if (_apiKey == null) return text;

    final cached = prefs.getString(cacheKey);
```

```
    if (cached != null) {
      return cached;
    }

    try {
      final url = Uri.parse(
'https://api.mymemory.translated.net/get?q=${Uri.encodeComponent(text)}'
            '&langpair=$from|$to'
            '&de=$_email'
            '&key=$_apiKey',
      );

      final response = await http.get(url);

      if (response.statusCode == 200) {
        final data = jsonDecode(response.body);
        final translatedText = data['responseData']['translatedText'];

        //Αποθηκεύει τη μετάφραση στο SharedPreferences
        await prefs.setString(cacheKey, translatedText);
        return translatedText;
      } else {
        print('❌ Σφάλμα MyMemory: ${response.statusCode} -
${response.body}');
      }
    } catch (e) {
      print('❌ Σφάλμα κατά την κλήση του API: $e');
    }

    return text;
  }
}
```