ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΩΝ ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πτυχιακή Εργασία

Τίτλος Πτυχιακής Εργασίας	Συγκριτική Ανάλυση Τεχνολογιών 2D Barcodes και QR Codes Comparative Analysis of 2D barcodes and QR codes
Ονοματεπώνυμο Φοιτητή	Δημήτριος Αναστασίου
Πατρώνυμο	Ιωάννης
Αριθμός Μητρώου	Π/ 20015
Επιβλέπων	Ευάγγελος Σακκόπουλος, Καθηγητής

Ημερομηνία Παράδοσης Ιούλιος 2026

Copyright ©

ΠΕΡΙΕΧΟΜΕΝΟ

1. Εισαγωγή και Σκοπός της ‘Ερευνας 7
1.1. Αντικείμενο της Εργασίας 7
1.2. Μεθοδολογία της Έρευνας 7
1.3 Ιστορική Αναδρομή 8
2. Θεωρητικό Υπόβαθρο – Τύποι Barcodes 9
2.1. Μονοδιάστατο (1D) Barcodes – Γραμμικά Barcodes 9
2.2 Δισδιάστατα (2D) Barcodes — Matrix Codes 11
3. Αναλυτική Σύγκριση 2D Barcodes 14
3.1. Σύγκριση Βασικών Χαρακτηριστικών 14
3.1. Σύγκριση Βασικών Χαρακτηριστικών 16
3.2 Αλγόριθμος Διόρθωσης Σφαλμάτων Reed-Solomon 17
4. Οδηγίες Αποτελεσματικής Χρήσης QR Codes 18
4.1 Βασικές Χρήσεις QR Codes 18
4.2 Οι 13 Οδηγίες Χρηστικότητας (NN/g) — Αναλυτική Περιγραφή 19
4.3 Ασφάλεια QR Codes και Κίνδυνοι 21
4.3 Η Επίδραση της Πανδημίας COVID-19 στην Υιοθέτηση QR Codes 22
5. Μέγεθος, Διαστάσεις και Ανάλυση Εκτύπωσης QR Codes 23
5.1 Εκδόσεις (Versions) QR Code – Αρχιτεκτονική Modules 23
5.2 Παράγοντες που Επηρεάζουν το Μέγεθος 23
5.3 Ελάχιστα και Ιδανικά Μεγέθη 24
5.4 Σχέση Μεγέθους – Αναγνωσιμότητας – Modules 24
5.5 Κανόνας Αναλογίας 10:1 (Distance-to-Size) 25
5.6 Quiet Zone 25
5.7 Βελτιστοποίηση Μεγέθους 26
5.8 Υπολογισμός DPI για Εκτύπωση 26

6. Βιβλιοθήκες Ανοιχτού Κώδικα — Αναλυτική Αξιολόγηση 27

6.1 Μεθοδολογία Αξιολόγησης 27

6.2 Python Βιβλιοθήκες 27

6.2.1 qrcode (Python) — Εκτενής Ανάλυση 27
6.2.2 OpenCV (Python) — Εκτενής Ανάλυση 28
6.2.3 ZBar (pyzbar) — Εκτενής Ανάλυση 29

6.3 Java Βιβλιοθήκες 29

6.3.1 ZXing (Java) — Εκτενής Ανάλυση 29
6.3.2 BoofCV (Java) — Εκτενής Ανάλυση 30

6.4 Kotlin Βιβλιοθήκες 31

6.4.1 ZXing (Kotlin) — Εκτενής Ανάλυση 31
6.4.2 Code Scanner (Kotlin/Android) — Εκτενής Ανάλυση 31

6.5 Σύγκριση Βιβλιοθηκών 31

7. Δημιουργία Παραδειγμάτων ανά Βιβλιοθήκη — Πρακτική Υλοποίηση 32

8. Πλατφόρμα Δοκιμών — React Web Application 33

8.1 Σκοπός του Web Application 33

8.2 Τεχνολογικό Stack 33

8.3 Αρχιτεκτονική Εφαρμογής 34

8.4 Τρόπος Λειτουργίας και Testing 35

9. Προχωρημένη Ανάλυση — Κωδικοποίηση Δεδομένων σε Πολλαπλές Μορφές Barcode 35

9.1 Σκοπός 35

9.2 Datasets 36

9.3 Διαδικασία Κωδικοποίησης 37

9.4 Υπολογισμός DPI 37

9.5 Δοκιμή Αναγνωσιμότητας (Decode Check) 38

9.6 Αποτελέσματα 38

9.6.1 JSON 1KB Dataset 38
9.6.2 ICAO DG1 (MRZ) Dataset 38
9.6.3 Image Dataset (1-3 KB) 39

9.7 Ανάλυση Αναγνωσιμότητας ανά Βιβλιοθήκη 39

10. Αρχιτεκτονική και Ανάλυση Κώδικα 40

10.1 Χρήση React 41

10.2 Συλλογή input για παραγωγή των QR 41

10.2.1 Παραγωγή των QR 43
10.2.2 Παράδειγμα Δημιουργίας QR Code σε Native Γλώσσα της βιβλιοθήκης 47
10.3 Advanced Analysis 48
11. Συμπεράσματα και Προτάσεις 52
11.1 Επιλογή Τύπου Barcode 52
11.2 Επιλογή Βιβλιοθήκης 53
11.3 Βέλτιστες Πρακτικές Σχεδιασμού 53
11.4 Περιορισμοί Κωδικοποίησης Εικόνων 54
11.5 Συμπέρασμα 54
12. Βιβλιογραφία 55
12.1 Ιστορία και Εξέλιξη Barcodes 55
12.2 Reed-Solomon Error Correction 55
12.3 Σύγκριση Τύπων 2D Barcodes 55
12.4 QR Code Usability, Μέγεθος και Εκτύπωση 56
12.5 Εφαρμογές σε Βιομηχανία και Μεταφορές 56
12.6 Βιβλιοθήκες Λογισμικού — Επίσημη Τεκμηρίωση 56
12.7 COVID-19 και Σύγχρονη Υιοθέτηση 57
12.8 Ασφάλεια QR Codes 57

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον καθηγητή Ευάγγελο Σακκόπουλο για την ευκαιρία που μου προσέφερε να συνεργαστούμε ώστε να ολοκληρωθεί η παρούσα πτυχιακή εργασία και να έρθουν εις πέρας οι σπουδές μου στο Τμήμα Πληροφορικής και το Πανεπιστήμιο Πειραιά. Θα ήθελα, επίσης, να ευχαριστήσω τους καθηγητές του τμήματος για τη πληθώρα γνώσεων που προσέφεραν απλόχερα και συνέβαλαν ώστε να τεθούν γερές βάσεις για τη μελλοντική μου πορεία και το επαγγελματικό γίγνεσθαι. Τέλος, δεν θα μπορούσα να μην ευχαριστήσω τους συμφοιτητές μου με τη βοήθεια των οποίων ολοκληρώσαμε μία σειρά ομαδικών εργασιών και καλύψαμε μαζί σημαντικό έδαφος στο ταξίδι των σπουδών του τμήματός μας.

1. Εισαγωγή και Σκοπός της ‘Ερευνας

1.1. Αντικείμενο της Εργασίας

Η παρούσα πτυχιακή εργασία πραγματεύεται τη μελέτη, ανάλυση και πρακτική εφαρμογή των τεχνολογιών μονοδιάστατων (1D) και δισδιάστατων (2D) barcodes, με κεντρικό άξονα τα QR Codes. Τα QR Codes αποτελούν σήμερα μία από τις πιο διαδεδομένες τεχνολογίες αναγνώρισης και σύλληψης δεδομένων με εφαρμογές που εκτείνονται στο εμπόριο, σε ταξιδιωτικά έγγραφα, σε ηλεκτρονικά εισιτήρια για τη ταυτοποίηση χρηστών και άλλα συναφή παραδείγματα όπου γίνεται χρήση των QR για σύλληψη σημαντικής πληροφορίας.

Ο στόχος της εργασίας έχει θεωρητικό, ερευνητικό και πρακτικό υπόβαθρο. Αρχικά, στοχεύει στη θεωρητική κατανόηση των διαφορών μεταξύ των διαφόρων τύπων barcodes – τόσο μονοδιάστατων όσο και δισδιάστατων. Στη συνέχεια, εμβαθύνει στην εύρεση, αξιολόγηση και δοκιμή βιβλιοθηκών ανοιχτού κώδικα σε τρεις γλώσσες προγραμματισμού (Python, Java, Kotlin) καθώς και σε JavaScript μέσω web application. Τέλος, όσον αφορά το πρακτικό μέρος, έχει πραγματοποιηθεί κωδικοποίηση στοχεύοντας σε τρεις διαφορετικούς τύπους δεδομένων (JSON, ICAO DG1, Passport Dataset, μικρές εικόνες) σε πολλαπλές μορφές barcode (QR Code, PDF417, Aztec) παράγοντας και αναλύοντας τα αποτελέσματα από τη χρήση των βιβλιοθηκών ως προς τις διαστάσεις, της απαιτήσεις εκτύπωσης (DPI) και την αναγνωσιμότητα.

1.2. Μεθοδολογία της Έρευνας

Η μεθοδολογία που βασίστηκε σε πέντε διαδοχικές φάσεις με κάθε φάση να επηρεάζεται από τη προηγούμενη.

Φάση 1 – Βιβλιογραφική Επισκόπηση: Μελετήθηκαν εκτενώς τα θεωρητικά μέρη των τεχνολογιών barcode. Αναλύθηκαν οι διαφορές μεταξύ μονοδιάστατων και δισδιάστατων barcodes, καθώς επίσης και οι ιδιότητες κάθε τύπου δισδιάστατου barcode (QR Code, PDF417,

DataMatrix, Aztec). Βασικές πηγές της μελέτης αποτέλεσαν τα άρθρα των Dynamsoft, Scandit και Docutain SDK καθώς και τα αντίστοιχα πρότυπα ISO/IEC.

Φάση 2 – Εύρεση και Αξιολόγηση Βιβλιοθηκών: Αναζητήθηκαν βιβλιοθήκες ανοιχτού κώδικα σε Python, Java, Kotlin . Εντοπίστηκαν συνολικά 7 βιβλιοθήκες: qr code (Python), OpenCV (Python), ZBar/pyzbar (Python), ZXing (Java), BoofCV (Java), ZXing (Kotlin) και Code Scanner (Kotlin/Android). Κάθε βιβλιοθήκη μελετήθηκε ως προς τις δυνατότητές της για δημιουργία και ανάγνωση, την υποστήριξη χρωμάτων, τα επίπεδα διόρθωσης σφαλμάτων και την γενικότερη ευχρηστία.

Φάση 3 – Μελέτη Βέλτιστων Πρακτικών: Μελετήθηκαν οι οδηγίες αποτελεσματικής χρήσης QR Codes σύμφωνα με το Nielsen Norman Group, τα ζητήματα μεγέθους, ανάλυσης (resolution) και εκτύπωσης (DPI) σύμφωνα με το Triton Store, Uniqode και το ακαδημαϊκό άρθρο του IJARSCT.

Φάση 4 – Δημιουργία Προγραμματιστικών Παραδειγμάτων: Για κάθε βιβλιοθήκη που συλλέχθηκε αναπτύχθηκαν ολοκληρωμένα παραδείγματα κώδικα που κάλυψαν διαφορετικά error correction levels (L, M, Q, H), διαφορετικά μεγέθη (100x100, 200x200, 400x400) και διαφορετικούς χρωματικούς συνδυασμούς όσον αφορά το background του QR και τον χρωματισμό των modules.

Φάση 5 – Κωδικοποίηση και Ανάλυση: Κωδικοποιήθηκαν τρεις τύποι δεδομένων (1KB JSON, ICAO DG1 passport dataset, μικρή εικόνα 1-3KB) σε τρεις μορφές barcode (QR Code, PDF417, Aztec), με μέτρηση διαστάσεων, υπολογισμό DPI και δοκιμή αναγνωσιμότητας από πολλαπλές βιβλιοθήκες.

1.3 Ιστορική Αναδρομή

Το QR Code (Quick Response Code) εφευρέθηκε το 1994 από τον Masahiro Hara και την ομάδα του στη Denso Wave, θυγατρική εταιρεία του ομίλου Toyota. Η ανάγκη για τη δημιουργία του προέκυψε από τους περιορισμούς των παραδοσιακών μονοδιάστατων barcodes στη βιομηχανία αυτοκινήτων, τα οποία μπορούσαν να αποθηκεύσουν μόνο περίπου 20 αλφαριθμητικούς χαρακτήρες και απαιτούσαν πολλαπλές σαρώσεις ανά εξάρτημα (Denso Wave, n.d.).

Η σχεδίαση του QR Code βασίστηκε σε δύο βασικές αρχές: αφενός στη δυνατότητα αποθήκευσης σημαντικά περισσότερων δεδομένων — συμπεριλαμβανομένων χαρακτήρων Kanji και Kana — και αφετέρου στη δυνατότητα γρήγορης ανάγνωσης από πολλαπλές γωνίες. Τα τρία χαρακτηριστικά finder patterns (τα τετράγωνα στις τρεις γωνίες) εμπνεύστηκαν από τη σκακιέρα Go, ενώ η αναλογία 1:1:3:1:1 που χρησιμοποιείται σε αυτά επιλέχθηκε επειδή αποτελεί το λιγότερο κοινό μοτίβο σε τυπωμένα έγγραφα, εξασφαλίζοντας ότι ο σαρωτής μπορεί να εντοπίσει εύκολα τον κώδικα (Wikipedia, 2024).

Η Denso Wave κατέχει τα δικαιώματα πατέντας του QR Code, αλλά αποφάσισε στρατηγικά να μην τα ασκήσει, καθιστώντας την τεχνολογία ελεύθερα διαθέσιμη παγκοσμίως. Αυτή η απόφαση συνέβαλε καθοριστικά στην παγκόσμια υιοθέτησή του. Το 2000, η τεχνολογία εγκρίθηκε ως διεθνές πρότυπο (ISO/IEC 18004), ενώ η ευρεία δημόσια υιοθέτησή του ξεκίνησε στη δεκαετία του 2000 με την ενσωμάτωση αναγνωστών QR στα smartphones (Denso Wave, QRcode.com).

2. Θεωρητικό Υπόβαθρο – Τύποι Barcodes

2.1. Μονοδιάστατο (1D) Barcodes – Γραμμικά Barcodes

Τα μονοδιάστατα barcodes (linear barcodes) αποτελούν πλέον το παλαιότερο και πιο διαδεδομένο τύπο barcode. Περιέχουν μία σειρά πληροφοριών και διακρίνονται από παράλληλες γραμμές (bars) και κενά μεταξύ αυτών των γραμμών (spaces) μεταβλητού πλάτους. Η κωδικοποίηση των δεδομένων γίνεται, λοιπόν, σε ένα οριζόντιο άξονα το οποίο κατ' επέκταση σημαίνει ότι ο όγκος δεδομένων είναι περιορισμένος. Ωστόσο, τα μονοδιάστατα barcodes παραμένουν τα πιο χρησιμοποιούμενα barcodes λόγω της διαχρονικής χρήσης τους και της ευρείας αποδοχής παγκοσμίως (Docutain SDK 2024, Scandit 2024).

Σύμφωνα με τη Scandit, τα μονοδιάστατα barcodes μπορούν να αναγνωστούν τόσο με laser scanners όσο και με smart devices κάτι το οποίο δεν υποστηρίζεται πάντα από τα δισδιάστατα barcodes που απαιτούν συχνά συσκευές με κάμερα. Αυτή η συμβατότητα με παλαιότερης τεχνολογίας hardware αποτελεί σημαντικό πλεονέκτημα σε βιομηχανικά περιβάλλοντα όπου η αντικατάσταση της υπάρχουσας τεχνολογίας φαντάζει, συχνά, δύσκολη.

Αναλυτική Παρουσίαση κύριων 1D barcode τύπων:

- **EAN (European Article Number):** Ίσως ο πιο γνωστός τύπος παγκοσμίως, κωδικοποιεί 13 ή 8 ψηφία και χρησιμοποιείται στο εμπόριο και τις βιβλιοθήκες. Βασίζεται σ' ένα σύστημα κωδικών χώρας που επιτρέπει τη παγκόσμια μοναδικότητα κάθε προϊόντος (Docutain SDK, 2024).

- **UPS (Universal Product Code)**: Ο κύριος τύπος σε Βόρεια Αμερική, Αυστραλία, Ηνωμένο Βασίλειο και Νέα Ζηλανδία. Εδώ υπάρχουν δύο παραλλαγές, UPC-A με κωδικοποίηση 12 ψηφίων που χρησιμοποιείται στα περισσότερα προϊόντα και UPC-E με κωδικοποίηση 6 ψηφίων για μικρότερα προϊόντα. Συνδέεται άμεσα με το τύπο barcode EAN (Scandit, 2024).

- **Code128**: Ένας τύπος υψηλής πυκνότητας μπορεί να κωδικοποιήσει αλφαριθμητικά δεδομένα και ειδικούς χαρακτήρες. Χρησιμοποιείται ευρέως στον βιομηχανικό τομέα λόγω της ευελιξίας που το χαρακτηρίζει και του compact μεγέθους του (Docutain SDK, 2024).

- **Code39**: Αποτελεί αλφαριθμητικό barcode το οποίο είναι ικανό να κωδικοποιήσει κεφαλαία γράμματα και αριθμούς. Παρόμοιος με το Code128 αλλά λιγότερο compact και με χαμηλότερη πυκνότητα δεδομένων (Scandit, 2024).

- **ITF (Interleaved 2 of 5):** Αριθμητικός barcode υψηλής πυκνότητας που κωδικοποιεί ψηφία σε ζεύγη. Χρησιμοποιείται κυρίως για τη σήμανση κιβωτίων και παλετών στη logistics (Docutain SDK, 2024).

- **GS1 DataBar:** Σχεδιάστηκε για μικρά αντικείμενα στον τομέα υγείας και νωπών τροφίμων. Η εκδοχή GS1 DataBar Expanded μπορεί να κωδικοποιήσει έως 74 αριθμητικούς ή 41 αλφαριθμητικούς χαρακτήρες, επιτρέποντας πρόσθετες πληροφορίες όπως βάρος, ημερομηνία λήξης και κωδικούς κουπονιών (Docutain SDK, 2024).

- **Codabar, Code 93, Code 11, MSI Plessey:** Εξειδικευμένοι τύποι που χρησιμοποιούνται σε βιβλιοθήκες, τράπεζες αίματος, τηλεπικοινωνίες και διαχείριση αποθεμάτων αντίστοιχα (Docutain SDK, 2024).

2.2 Δισδιάστατα (2D) Barcodes — Matrix Codes

Μία βασική διαφορά των δισδιάστατων barcodes από τα μονοδιάστατα barcodes είναι το μέγεθος δεδομένων που υποστηρίζουν καθώς τα δισδιάστατα barcodes υποστηρίζουν κωδικοποίηση δεδομένων τόσο στον οριζόντιο όσο και στον κάθετο άξονα. Για να γίνει αντιληπτή αυτή η διαφορά μπορεί να γίνει μία αναδρομή στο αρχείο της Scandit σύμφωνα με την οποία ένας UPC-E είναι ικανός να κωδικοποιήσει μόλις 6 αλφαριθμητικούς χαρακτήρες ενώ ένας Aztec μπορεί να κωδικοποιήσει έως 3.067 αλφαριθμητικούς χαρακτήρες.

Ένα θεμελιώδες πλεονέκτημα των δισδιάστατων barcodes είναι οι ενσωματωμένοι αλγόριθμοι διόρθωσης σφαλμάτων (error correction level). Αυτοί οι αλγόριθμοι επιτρέπουν την ανάγνωση του barcode ακόμη κι αν αυτό έχει καταστραφεί, σκιστεί ή γρατσουνιστεί. Αυτό σημαίνει ότι υποστηρίζουν το γρήγορο σκανάρισμα και καθιστά τα δισδιάστατα barcodes κατάλληλα για εφαρμογές μεγάλου μεγέθους που απαιτούν σημαντικό όγκο δεδομένων.

Σύγκριση 1D και 2D Barcodes ανά χαρακτηριστικό:

- Κωδικοποίηση Δεδομένων: Τα μονοδιάστατα υποστηρίζουν κωδικοποίηση μόνο στον οριζόντιο άξονα ενώ τα δισδιάστατα υποστηρίζουν τόσο στον οριζόντιο όσο και στον κατακόρυφο άξονα.

- Πυκνότητα Δεδομένων: Τα μονοδιάστατα υποστηρίζουν 6 ψηφία σε UPC-E ενώ τα δισδιάστατα υποστηρίζουν έως και 3.067 χαρακτήρες.

- Αλγόριθμοι διόρθωσης σφαλμάτων: Υποστηρίζονται μόνο από τα δισδιάστατα barcodes.

- Απαιτήσεις σε hardware: Τα μονοδιάστατα απαιτούν laser scanners ή smart devices ενώ τα δισδιάστατα απαιτούν smart devices με κάμερα.

- Ανθεκτικότητα σε ζημιά: Τα μονοδιάστατα έχουν μικρή ανθεκτικότητα ενώ τα δισδιάστατα έχουν έως και 30% recovery.

- Παραδείγματα: Για τα μονοδιάστατα UPC, EAN, Code128, Code39 και για τα δισδιάστατα QR Code, Aztec, Data Matrix, PDF417.

Αναλυτική Παρουσίαση κύριων 2D Barcodes:

QR Code (Quick Response Code): Αποτελεί το πιο διάσημο δισδιάστατο barcode παγκοσμίως και δημιουργήθηκε το 1994 από τη Denso Wave. Σύμφωνα με τη Scandit (2024), όσον αφορά τους τρόπου κωδικοποίησης δεδομένων, υποστηρίζει αριθμητικό, αλφαριθμητικό, byte/binary καθώς και Kanji ενώ η χωρητικότητά του ανέρχεται σε 7.089 αριθμητικούς, 4.296 αλφαριθμητικούς χαρακτήρες, 1.817 χαρακτήρες Kanji ή 2.953 bytes δεδομένων. Τα QR Codes είναι δημόσια και ελεύθερα για χρήση, γι' αυτό και η τόσο διαδεδομένη χρήση τους. Επιπρόσθετα, υποστηρίζει τη διόρθωση σφαλμάτων κάνοντας χρήση του αλγορίθμου Reed-Solomon επιτρέποντας την ανάγνωση του QR ακόμη κι αν έχει καταστραφεί το 30% του barcode (Dynamsoft, 2024). Η χρήση του συναντάται κυρίως σε πληρωμές, marketing, mobile apps, loyalty programs και virtual stores. Τέλος, τυποποιούνται με ISO/IEEC 18004.

Micro QR Code: Μικρότερη έκδοση του κανονικού QR Code, σχεδιασμένη για εφαρμογές με περιορισμένο χώρο. Μπορεί να αποθηκεύσει έως 35 αριθμητικούς χαρακτήρες. Χρησιμοποιείται σε μικρά προϊόντα και συσκευασίες (Docutain SDK, 2024).

rMQR Code (Rectangular Micro QR): Ορθογώνια QR Code για εφαρμογές με περιορισμό στη χωρητικότητα. Μπορεί να κωδικοποιήσει έως και 361 αριθμητικούς χαρακτήρες και χρησιμοποιείται στο λιανεμπόριο, σε logistics και ηλεκτρονικά (Docutain SDK, 2024).

PDF417: Δισδιάστατο barcode το οποίο υποστηρίζει υψηλή χωρητικότητα δεδομένων και αποτελείται από πολλαπλές οριζόντιες γραμμές. Σε αντίθεση με τα QR και DataMatrix τα οποία είναι matrix barcodes, το PDF417 μπορεί να διαβαστεί γραμμικά από τα αριστερά προς τα δεξιά σύμφωνα με τη Dynamsoft (2024). Μπορεί να αποθηκεύσει έως 1,1KB δεδομένων, υποστηρίζει την αναγνωσιμότητα ακόμη κι αν έχει καταστραφεί το 50% του barcode με χρήση του αλγορίθμου Reed-Solomon και είναι, επίσης, δημόσια και ελεύθερα για χρήση. Η χρήση του συναντάται σε boarding passes αεροπορικών εισιτηρίων, ταυτότητες, διπλώματα οδήγησης και postal packages (Scandit 2024, Dynamsoft 2024). Τέλος, τυποποιούνται με ISO/IEC 15438.

DataMatrix: Δισδιάστατα matrix barcodes υψηλής πυκνότητας όπου η υψηλή πυκνότητα προσφέρει το πλεονέκτημα της κάλυψης μικρότερου χώρου στα προϊόντα. Υποστηρίζει τη κωδικοποίηση 2.335 αλφαριθμητικών χαρακτήρων, 3.116 αριθμητικών ή 1.556 bytes. Η εκδοχή ECC 220 χρησιμοποιεί τον αλγόριθμο Reed-Solomon ενώ παλαιότερες εκδόσεις όπως το ECC 000-140 χρησιμοποιούν convolutional error correction (Dynamsoft 2024). Η US Electronic Industries Alliance (EIA) συνιστά τη χρήση τους για τη σήμανση μικρών ηλεκτρονικών εξαρτημάτων. Κύριες χρήσεις DataMatrix συναντάται σε direct parts marking, ηλεκτρονικά, φαρμακευτικά και συσκευασίες (Scandit 2024). Τέλος, τυποποιούνται με ISO/IEC 16022.

Aztec Code: Δισδιάστατα barcodes τα οποία, σε αντίθεση με τα υπόλοιπα, δεν απαιτούν quiet zone (κενό χώρο γύρω από το κώδικα). Μπορεί να κωδικοποιήσει έως 3.067 αλφαριθμητικούς, 3.882 αριθμητικούς ή 1.914 bytes. Συναντάται σε εισιτήρια μεταφορών, boarding passes, έγγραφα υγείας και κύριο πλεονέκτημά τους αποτελεί η αναγνωσιμότητα σε κακή ανάλυση με αποτέλεσμα εκτυπωμένα έγγραφα ή εισιτήρια που εκτυπώνονται με χαμηλή ποιότητα ή παρουσιάζονται με οθόνη κινητού να παραμένουν αναγνώσιμα. Τέλος, πιστοποιούνται με ISO/IEC 24778.

3. Αναλυτική Σύγκριση 2D Barcodes

3.1. Σύγκριση Βασικών Χαρακτηριστικών

Με βάση τη μελέτη της Dynamsoft (2024) και της Scandit (2024) παρουσιάζεται παρακάτω η αναλυτική σύγκριση ανάμεσα σε QR Code, PDF417, DataMatrix και Aztec χαρακτηριστικό προς χαρακτηριστικό.

1. Πυκνότητα Δεδομένων

    - QR Code: Υψηλή
    - PDF417: Υψηλή
    - DataMatrix: Υψηλή
    - Aztec: Υψηλή

2. Character Set

    - QR Code: Full ASCII + Kanji
    - PDF417: Full ASCII
    - DataMatrix: Full ASCII
    - Aztec: Full ASCII

3. Μέγιστοι Αριθμητικοί Χαρακτήρες

    - QR Code: 7.089
    - PDF417: 2.710
    - DataMatrix: 3.116
    - Aztec: 3.832

4. Μέγιστοι Αλφαριθμητικοί Χαρακτήρες

    - QR Code: 4.296
    - PDF417: 1.850

- DataMatrix: 2.335
- Aztec: 3.067

5. Μέγιστα Bytes Δεδομένων

    - QR Code: 2.953
    - PDF417: 1.108
    - DataMatrix: 1.556
    - Aztec: 1.914

6. Γεωμετρικό Σχήμα

    - QR Code: Τετράγωνο
    - PDF417: Ορθογώνιο
    - DataMatrix: Τετράγωνο
    - Aztec: Τετράγωνο

7. Αλγόριθμος EC

    - QR Code: Reed-Solomon
    - PDF417: Reed-Solomon
    - DataMatrix: Reed-Solomon (ECC 200)
    - Aztec: Reed-Solomon

8. Μέγιστη Ανάκτηση EC

    - QR Code: 30%
    - PDF417: 50%
    - DataMatrix: Configurable
    - Aztec: Configurable

9. Quiet Zone Required

    - QR Code: Ναι
    - PDF417: Ναι
    - DataMatrix: Ναι

- Aztec: Όχι

10. Τρόπος Ανάγνωσης

- QR Code: 2D imager
- PDF417: 2D imager
- DataMatrix: 2D imager
- Aztec: 2D imager

11. Υποστήριξη Kanji

- QR Code:
- PDF417:
- DataMatrix:
- Aztec:

12. Public Domain

- QR Code: Ναι
- PDF417: Όχι
- DataMatrix: Όχι
- Aztec: Όχι

13. ISO Standard

- QR Code: ISO/IEC 18004
- PDF417: ISO/IEC 15438
- DataMatrix: ISO/IEC 16022
- Aztec: ISO/IEC 24778

3.1. Σύγκριση Βασικών Χαρακτηριστικών

QR Code: Ιδανικό για mobile εφαρμογές, online app πληρωμές, marketing καμπάνιες, virtual stores, website login, σύνδεση σε mobile operating systems. Η μεγάλη χωρητικότητα και η

ευκολία αναγνωσιμότητας που το διακρίνουν το καθιστούν τον πιο δημοφιλή τύπο σύμφωνα με τη Dynamsoft (2024).

PDF417: Ιδανικό για boarding passes αεροπορικών εταιριών, διπλώματα οδήγησης, ταυτότητες, postal packages. Η δυνατότητα γραμμικής ανάγνωσης και η ανθεκτικότητα σε σφάλματα (50%) το καθιστούν ιδανικό και για κυβερνητικά έγγραφα σύμφωνα με τη Scandit (2024).

DataMatrix: Ιδανικό για μικρά εξαρτήματα, ηλεκτρονικά προϊόντα, φαρμακευτικά και βιομηχανική σήμανση. Το μαζεμένο μέγεθος και η ικανότητα ανάγνωσης σε χαμηλή ανάλυση το καθιστούν ιδανικό για μικρά αντικείμενα σύμφωνα με τη Scandit (2024) .

Aztec Code: Ιδανικό για εισιτήρια μεταφορών, billing και healthcare.Σύμφωνα με τη Scandit (2024), Η ανάγνωση χωρίς την απαίτηση quiet zone και η αντοχή σε κακή ανάλυση εκτύπωσης το καθιστούν ιδανικό για εισιτήρια που εμφανίζονται σε κινητά ή εκτυπώνονται σε χαμηλή ποιότητα.

3.2 Αλγόριθμος Διόρθωσης Σφαλμάτων Reed-Solomon

Όπως αναφέρθηκε στη σύγκριση, και τα τέσσερα πρότυπα 2D barcodes (QR Code, PDF417, DataMatrix, Aztec) χρησιμοποιούν τον αλγόριθμο Reed-Solomon για τη διόρθωση σφαλμάτων. Ο αλγόριθμος αυτός παρουσιάστηκε αρχικά το 1960 από τους Irving S. Reed και Gustave Solomon στο σεμινάριο τους «Polynomial Codes Over Certain Finite Fields» (Reed & Solomon, 1960).

Ο αλγόριθμος λειτουργεί αντιμετωπίζοντας τα δεδομένα ως συντελεστές ενός πολυωνύμου πάνω σε ένα πεπερασμένο πεδίο (Galois Field). Στη διαδικασία κωδικοποίησης, υπολογίζονται πρόσθετα σύμβολα ισοτιμίας (parity symbols) μέσω αριθμητικής Galois Field, τα οποία προσαρτώνται στο αρχικό μήνυμα. Κατά την αποκωδικοποίηση, ο σαρωτής μπορεί να χρησιμοποιήσει αυτά τα σύμβολα για να εντοπίσει και να ανακατασκευάσει μαθηματικά τα αρχικά δεδομένα, ακόμα και αν μέρος του barcode λείπει ή είναι κατεστραμμένο.

Στα QR Codes συγκεκριμένα, τα δεδομένα χωρίζονται σε blocks και κάθε block συνοδεύεται από Reed-Solomon codewords σύμφωνα με το πρότυπο ISO/IEC 18004. Τα τέσσερα επίπεδα

διόρθωσης (L: ~7%, M: ~15%, Q: ~25%, H: ~30%) καθορίζουν πόση πλεονάζουσα πληροφορία προστίθεται.

Η χρήση Reed-Solomon κωδίκων καθιστά τα QR codes ιδιαίτερα ανθεκτικά σε «burst errors» — δηλαδή γρατσουνιές, μερική κάλυψη ή φυσική φθορά — γεγονός κρίσιμο για χρήση σε πραγματικές συνθήκες. Επιπλέον, η δυνατότητα ανάκτησης δεδομένων επιτρέπει την ενσωμάτωση λογοτύπων ή χρωμάτων μέσα στο QR code, αφού ο αλγόριθμος αντιμετωπίζει τις τροποποιήσεις ως «σφάλματα» και τα αγνοεί κατά την αποκωδικοποίηση (Math and Tech.).

4. Οδηγίες Αποτελεσματικής Χρήσης QR Codes

4.1 Βασικές Χρήσεις QR Codes

Σύμφωνα με τη Nielsen Norman Group τα QR Codes εξυπηρετούν τη σταδιακή αποκάλυψη πληροφοριών (Progressive Disclosure), τις μεταβάσεις μεταξύ καναλιών αλληλεπίδρασης (Channel Transitions) και τη μείωση αναγκών εκτύπωσης. Πιο αναλυτικά:

Proggresive Disclosure: Τα QR codes μπορούν να παρέχουν πρόσβαση σε πρόσθετες πληροφορίες που δεν χωρούν σε περιορισμένους φυσικούς χώρους ή δεν αφορούν όλους τους χρήστες. Λειτουργούν ως τα «Learn more links» του φυσικού κόσμου. Για παράδειγμα, μια μικρή πινακίδα που προωθεί μια τοπική πρωτοβουλία δεν χωρά πολλές πληροφορίες, αλλά ένα QR code στη γωνία επιτρέπει πρόσβαση σε περισσότερες λεπτομέρειες. Ο αριθμός σαρώσεων αυτού του QR code μπορεί να αποκαλύψει σημαντικά insights σχετικά με το ενδιαφέρον του κοινού (NN/g, 2024).

Channel Transitions: Η πιο συχνή χρήση είναι η μετάβαση από τον φυσικό στον ψηφιακό χώρο. Για παράδειγμα, αρκετά συχνά τα μενού εστιατορίων δίνονται στους πελάτες μέσω QR Code που είναι τοποθετημένα στα τραπέζια, ο χρήστης σκανάρει το QR και μεταβαίνει στο ψηφιακό κόσμο του τηλεφώνου του όπου μπορεί να αποκτήσει πρόσβαση στο μενού του

καταστήματος. Ένα άλλο παράδειγμα που εντάσσεται στο channel transition είναι η επικοινωνία μεταξύ ψηφιακών συσκευών (τηλεοράσεων, κινητών, υπολογιστών κ.ά.) όπου παρατηρούνται μεταβάσεις από τη μία συσκευή στην άλλη με σκοπό τη κοινοποίηση ενός passkey ή άλλων μορφών authentication.

Μείωση Αναγκών Εκτύπωσης: Τα QR codes παρέχουν τρόπο μεταφοράς εντύπων πληροφοριών στον ψηφιακό χώρο. Αυτό προσφέρει καθαριότητα, ευκολία και περιβαλλοντικά οφέλη. Χαρακτηριστικό παράδειγμα: ένα QR code που συνδέει στο μενού εστιατορίου μειώνει τη διασπορά μικροβίων, εξαλείφει την ανάγκη εκτύπωσης μενού και επιτρέπει σε απεριόριστο αριθμό ατόμων να έχουν πρόσβαση ταυτόχρονα (NN/g, 2024).

4.2 Οι 13 Οδηγίες Χρηστικότητας (NN/g) — Αναλυτική Περιγραφή

Η Nielsen Norman Group έχει δημοσιεύσει 13 αναλυτικές οδηγίες για την αποτελεσματική χρήση QR codes. Οι οδηγίες αυτές αποτελούν το de facto industry standard για τη σχεδίαση QR code experiences:

Οδηγία 1 — Ενημέρωση χρηστών για τη λειτουργία του QR code: Σε αντίθεση με τα links με ετικέτες ή τα URLs που περιέχουν πληροφορίες για τον προορισμό, τα QR codes δεν έχουν κανένα «information scent» από μόνα τους. Δεν λένε τίποτα στον χρήστη για το πού οδηγούν. Οι σχεδιαστές πρέπει να παρέχουν επαρκή πληροφορία: σύντομο URL, ετικέτα, ή branding. Ωστόσο, ένα λογότυπο μόνο του δεν αρκεί. Η πληροφορία πρέπει να είναι τόσο ορατή και κατανοητή όσο και ο ίδιος ο κώδικας (NN/g, 2024).

Οδηγία 2 — Διευκρίνιση συμβατών συσκευών: Αν τα QR codes πρέπει να σαρωθούν μέσα σε συγκεκριμένη εφαρμογή ή από συγκεκριμένη συσκευή, αυτή η πληροφορία πρέπει να παρουσιάζεται δίπλα στον κώδικα. Για παράδειγμα, στην Κίνα, ορισμένα QR codes σε τραπέζια εστιατορίων πρέπει να σαρωθούν με QWeChat ή Alipay (NN/g, 2024).

Οδηγία 3 — Mobile-friendly σελίδες: Μπορούμε να υποθέσουμε ότι οι χρήστες που σαρώνουν QR code θα επισκεφθούν τη σελίδα σε κινητή συσκευή. Τα QR codes πρέπει να οδηγούν σε mobile-dedicated ή responsive sites (NN/g, 2024).

Οδηγία 4 — Deep Linking: Οι χρήστες αναμένουν ότι το QR code θα τους οδηγήσει σε πληροφορίες άμεσα σχετικές με το context του κώδικα, και όχι σε μια γενική αρχική σελίδα. Η σάρωση που υπόσχεται μια ενέργεια αλλά καταλήγει στην αρχική σελίδα είναι ενοχλητική (NN/g, 2024).

Οδηγία 5 — Χρήση direct links αντί QR codes στο ίδιο κινητό: Αν η πληροφορία προορίζεται για πρόσβαση στην ίδια συσκευή που εμφανίζει τον κώδικα, ένα απευθείας link είναι πιο εύκολο. Τα QR codes σε mobile interfaces πρέπει να εμφανίζονται μόνο αν προορίζονται για σάρωση από άλλη συσκευή (π.χ. ψηφιακές κάρτες επιβίβασης) (NN/g, 2024).

Οδηγία 6 — Μη αντιστροφή χρωμάτων: Τα QR codes πρέπει να παρουσιάζονται σε light mode — σκούρο foreground σε ανοιχτό background. Τα σκούρα χρώματα απορροφούν περισσότερο φως, δημιουργώντας πιο καθαρά contours για τις τεχνολογίες σάρωσης. Αυτό είναι ιδιαίτερα σημαντικό σε ακραίες συνθήκες φωτισμού, όπως σκοτεινά δωμάτια ή άμεσο ηλιακό φως. Αν και τα περισσότερα σύγχρονα τηλέφωνα μπορούν να σαρώσουν inverted QR codes, δεν μπορούν όλες οι τεχνολογίες φυσικών τοποθεσιών (NN/g, 2024). Αυτή η παρατήρηση επιβεβαιώθηκε κατά τη δοκιμή — τα inverted QR codes αναγνωρίστηκαν με δυσκολία από ορισμένες βιβλιοθήκες.

Οδηγία 7 — Authentication μεταξύ συσκευών: Τα QR codes παρέχουν ισχυρό μέσο για τους χρήστες που είναι ήδη συνδεδεμένοι σε λογαριασμό σε κινητή συσκευή να αυθεντικοποιηθούν εύκολα σε άλλη συσκευή (smart TV, υπολογιστή, άλλο smartphone) (NN/g, 2024).

Οδηγία 8 — Progressive disclosure σε φυσικά προϊόντα: Τα QR codes είναι ιδιαίτερα χρήσιμα για απόκρυψη πρόσθετων λεπτομερειών σε προϊόντα με περιορισμένο χώρο εμφάνισης — διαφημίσεις, σήμανση, συσκευασίες. Πρέπει να προσφέρουν πλούσια πληροφορία, και όχι γενικές ετικέτες «Learn more» (NN/g, 2024).

Οδηγία 9 — Ενημερωμένες πληροφορίες: Τα QR codes που βρίσκονται σε φυσικές εκτυπώσεις δεν μπορούν να ανακληθούν. Ακόμα και αν ο κώδικας αφορούσε προσωρινή

πληροφορία, κάποιοι χρήστες μπορεί να τον σαρώσουν πολύ αργότερα. Παλιοί κώδικες πρέπει να ανακατευθύνουν σε σχετική σελίδα που εξηγεί ότι η ισχύς έχει παρέλθει (NN/g, 2024).

Οδηγία 10 — Χρόνος σάρωσης >15 δευτερόλεπτα: Ο χρήστης χρειάζεται τουλάχιστον 15 δευτερόλεπτα για να σαρώσει ένα QR code (παρατηρεί, αξιολογεί, αποφασίζει, βγάζει κινητό, ανοίγει κάμερα, σκοπεύει, εστιάζει, πατά στο link. Σε γρήγορες-εκθέσεις, ένα σύντομο, ομοιόμορφο URL λειτουργεί καλύτερα (NN/g, 2024).

Οδηγία 11 — Μη αποκλειστική εξάρτηση: Τα QR codes δεν μπορούν να απομνημονευτούν. Αν η πρόσβαση σε κάποια σελίδα εξαρτάται αποκλειστικά από QR code, οι χρήστες δεν θα μπορούν να ξαναεπισκεφτούν τη σελίδα χωρίς τον κώδικα (NN/g, 2024).

Οδηγία 12 — Τοποθέτηση κοντά στο σχετικό περιεχόμενο: Σύμφωνα με την αρχή Gestalt της εγγύτητας (proximity), τα αντικείμενα κοντά μεταξύ τους θεωρούνται σχετικά. Ο κώδικας πρέπει να τοποθετείται κοντά στο περιεχόμενο με το οποίο σχετίζεται, και όχι σε απομονωμένη γωνία (NN/g, 2024).

Οδηγία 13 — Επαρκές μέγεθος: Ακόμα και αν πολλές κάμερες τηλεφώνων διαθέτουν zoom, η εγγύτητα παραμένει ουσιαστική. Το ελάχιστο μέγεθος σύμφωνα με τα επίσημα standards είναι 1 cm × 1 cm, αλλά αυτό μπορεί να είναι πολύ μικρό. Η NN/g συνιστά ελάχιστο 2 cm × 2 cm. Κανόνας: για κάθε 10 cm απόστασης, το μέγεθος αυξάνεται κατά 1 cm. Στα 50 cm, ο κώδικας πρέπει να είναι τουλάχιστον 5 cm × 5 cm (NN/g, 2024).

4.3 Ασφάλεια QR Codes και Κίνδυνοι

Παρά τα πλεονεκτήματά τους, τα QR codes ενέχουν σημαντικούς κινδύνους ασφαλείας. Η βασική ευπάθεια έγκειται στο ότι οι χρήστες δεν μπορούν να δουν τον προορισμό ενός QR code πριν το σαρώσουν — σε αντίθεση με ένα URL που είναι ορατό και αξιολογήσιμο. Αυτή η «αδιαφάνεια» εκμεταλλεύεται η επίθεση γνωστή ως «quishing» (QR phishing).

Βασικοί Κίνδυνοι:

Phishing URLs: Οι επιτιθέμενοι δημιουργούν QR codes που ανακατευθύνουν σε ψεύτικες ιστοσελίδες σχεδιασμένες να υποκλέψουν οικονομικά στοιχεία ή προσωπικά δεδομένα.

Φυσική παραποίηση: Μία συνήθης επίθεση περιλαμβάνει την τοποθέτηση κακόβουλων QR code stickers σε δημόσιους χώρους — π.χ. σε μενού εστιατορίων ή αφίσες.

Πέρα από URLs: Τα QR codes μπορούν να κωδικοποιήσουν Wi-Fi credentials, SMS μηνύματα, geolocation δεδομένα, ή ακόμα και εντολές cryptocurrency transactions, διευρύνοντας τις δυνατότητες επίθεσης.

Κάποια μέτρα που μπορούν να ληφθούν:

Πρώτον, ο έλεγχος πηγής — αποφυγή σάρωσης QR codes σε μη αξιόπιστες τοποθεσίες. Δεύτερον, η επιθεώρηση URL πριν το άνοιγμα — τα σύγχρονα smartphones εμφανίζουν preview του URL. Τρίτον, η χρήση native camera apps αντί τρίτων εφαρμογών QR scanner. Τέλος, η πρόσφατη έρευνα προτείνει self-authenticating QR codes (SDMQR) που ενσωματώνουν κρυπτογραφικές υπογραφές για επαλήθευση αυθεντικότητας πριν ο χρήστης ακολουθήσει τον σύνδεσμο (Focardi & Luccio, 2019).

4.3 Η Επίδραση της Πανδημίας COVID-19 στην Υιοθέτηση QR Codes

Η πανδημία COVID-19 αποτέλεσε τον κυριότερο καταλύτη για την παγκόσμια υιοθέτηση των QR codes, μετατρέποντάς τα από εξειδικευμένο εργαλείο marketing σε καθημερινή ανάγκη.

Η κύρια κινητήρια δύναμη ήταν η ανάγκη ελαχιστοποίησης φυσικής επαφής με κοινόχρηστες επιφάνειες — μετρητά, μενού, έντυπα — για τη μείωση κινδύνου μετάδοσης. Τα smartphones, που ήδη υποστηρίζουν σάρωση QR μέσω native camera, εξάλειψαν το εμπόδιο εγκατάστασης ειδικών εφαρμογών.

Στον τομέα υγείας, τα QR codes χρησιμοποιήθηκαν για contactless check-ins σε νοσοκομεία, προγραμματισμό εμβολιασμών, πιστοποιητικά εμβολιασμού (vaccine passports) και ιχνηλάτηση επαφών (contact tracing). Στον τομέα πληρωμών, παρατηρήθηκε εκρηκτική αύξηση QR-based πληρωμών παγκοσμίως, με χώρες όπως η Κίνα και η Ινδία να καταγράφουν τεράστια αύξηση χρηστών mobile payments (Juniper Research, 2024).

Η μεταπανδημική εποχή δείχνει ότι τα QR codes δεν αποτελούν προσωρινή λύση αλλά μόνιμο στρατηγικό εργαλείο ψηφιακού μετασχηματισμού — από τον έλεγχο αυθεντικότητας προϊόντων μέχρι τα ψηφιακά «διαβατήρια προϊόντων» (digital product passports).

5. Μέγεθος, Διαστάσεις και Ανάλυση Εκτύπωσης QR Codes

5.1 Εκδόσεις (Versions) QR Code – Αρχιτεκτονική Modules

Σύμφωνα με τη Uniqode (2024), ένα QR Code αποτελείται από μία σειρά τετραγώνων που ονομάζονται modules. Τα modules αυτά με τη σειρά τους σχηματίζουν μοτίβα που κωδικοποιούν πληροφορίες και κάθε module μπορεί να είναι σκούρο ή ανοιχτό. Τα QR Codes ταξινομούνται σε 40 εκδόσεις ξεκινώντας από το version 1 (21 x 21 modules = 441 modules) και φτάνοντας μέχρι το 40 (177 x 177 modules = 31.329 modules).

Η πρώτη έκδοση (Version 1) αποτελείται από 21 × 21 modules. Η Version 2 από 25 × 25, η Version 3 από 29 × 29, η Version 4 από 33 × 33. Με κάθε νέα έκδοση, προστίθενται 4 modules οριζόντια και κατακόρυφα. Η τελευταία, Version 40, φτάνει τα 177 × 177 modules. Ο γενικός τύπος είναι: Modules = 17 + 4 × Version (Uniqode, 2024).

Η επιλογή version εξαρτάται από τον όγκο δεδομένων: περισσότερα δεδομένα απαιτούν μεγαλύτερη version, δηλαδή μεγαλύτερο QR code. Ο τρόπος κωδικοποίησης (numeric, alphanumeric, byte, kanji) επηρεάζει επίσης το μέγεθος, καθώς κάθε mode έχει διαφορετικές απαιτήσεις χώρου (Uniqode, 2024).

5.2 Παράγοντες που Επηρεάζουν το Μέγεθος

Σύμφωνα με το Triton Store (2023), τρεις βασικοί παράγοντες καθορίζουν το μέγεθος ενός QR code:

Μήκος κωδικοποιημένων δεδομένων: Ο σημαντικότερος παράγοντας. Περισσότερα δεδομένα σημαίνουν μεγαλύτερο QR code. Η μέγιστη χωρητικότητα είναι 2.953 bytes, 4.296 αλφαριθμητικοί, 7.089 αριθμητικοί ή 1.817 Kanji χαρακτήρες (JIS X 0208) (Triton Store, 2023).

Επίπεδο Διόρθωσης Σφαλμάτων (ECL): Υπάρχουν τέσσερα επίπεδα: Level L (Low, 7%), Level M (Medium, 15%), Level Q (Quartile, 25%), Level H (High, 30%). Υψηλότερο ECL σημαίνει μεγαλύτερη ανθεκτικότητα αλλά και μεγαλύτερο μέγεθος, καθώς προστίθενται περισσότερα data modules με redundant πληροφορία (Triton Store, 2023).

Προσαρμογή σχεδίασης: Λογότυπα, γραφικά, σχήματα και χρώματα αυξάνουν το μέγεθος του QR code. Αυτά τα στοιχεία προσθέτουν αισθητική αξία αλλά καταλαμβάνουν χώρο (Triton Store, 2023).

5.3 Ελάχιστα και Ιδανικά Μεγέθη

Ελάχιστο πρακτικό μέγεθος: 2 cm × 2 cm (0.8" × 0.8") εξαιρουμένης της quiet zone. Μικρότερα QR codes μπορεί να λειτουργήσουν αν η απόσταση σάρωσης είναι κάτω από 5 cm, όπως σε επαγγελματικές κάρτες και εισιτήρια (Triton Store, 2023).

Ελάχιστα pixels: 38 × 38 pixels (minimum), 76 × 76 pixels (συνιστώμενο, αφού 1 cm ≈ 38 pixels) (Triton Store, 2023). Σύμφωνα με το Uniqode (2024), αυτό το threshold είναι και βάση αναφοράς: 38 × 38 px ≈ 1 cm × 1 cm.

Ιδανικό ψηφιακό μέγεθος: 240 × 240 pixels (6.35 cm × 6.35 cm) με ελάχιστο 72 DPI ανάλυση (Triton Store, 2023).

5.4 Σχέση Μεγέθους – Αναγνωσιμότητας – Modules

Σύμφωνα με το Uniqode (2024), η σχέση μεταξύ μεγέθους QR code και configuration modules είναι κρίσιμη. Με περισσότερα modules οριζόντια/κατακόρυφα, το QR code πρέπει να είναι μεγαλύτερο ώστε η σαρωτική συσκευή να μπορεί να διακρίνει κάθε module ξεχωριστά. Αν ένα

QR code 177 × 177 modules εκτυπωθεί σε πολύ μικρό μέγεθος, η διάκριση των modules γίνεται αδύνατη. Αντίστοιχα, αν ένα 11 × 11 module QR code μεγεθυνθεί υπερβολικά, τα modules μπορεί να γίνουν θολά ή τεντωμένα (stretched), βλάπτοντας τη σαρωσιμότητα.

Σύσταση SVG: Για αλλαγή μεγέθους χωρίς απώλεια ποιότητας, συνιστάται η χρήση του SVG format. Τα QR codes σε SVG μπορούν να κλιμακωθούν σε οποιοδήποτε μέγεθος χωρίς pixelation ή stretching (Uniqode, 2024).

5.5 Κανόνας Αναλογίας 10:1 (Distance-to-Size)

Ο κανόνας για τον υπολογισμό του ιδανικού μεγέθους QR code βάσει απόστασης σάρωσης είναι (Uniqode, 2024):

QR Code Size = Scanning Distance / 10

Δηλαδή, το μέγεθος του QR code πρέπει να είναι τουλάχιστον 1/10 της αναμενόμενης απόστασης σάρωσης.

Για επαγγελματικές κάρτες αν η απόσταση σάρωσης θεωρηθεί ότι είναι μεταξύ 10 και 15 εκατοστά τότε το ελάχιστο μέγεθος του QR πρέπει να είναι 1 με 1.5 εκατοστά. Για φυλλάδια αν η απόσταση σάρωσης θεωρηθεί ότι είναι μεταξύ 20 και 30 εκατοστά τότε το ελάχιστο μέγεθος του QR πρέπει να είναι 2 με 3 εκατοστά. Για αφίσες αν η απόσταση σάρωσης θεωρηθεί ότι είναι μεταξύ 100 και 200 εκατοστά τότε το ελάχιστο μέγεθος του QR πρέπει να είναι 10 με 20 εκατοστά. Για ψηφιακές οθόνες αν η απόσταση σάρωσης θεωρηθεί ότι είναι μεταξύ 30 και 50 εκατοστά τότε το ελάχιστο μέγεθος του QR πρέπει να είναι 240 x 240 px.

5.6 Quiet Zone

Η «quiet zone» είναι ο κενός χώρος γύρω από το QR code που επιτρέπει στη σαρωτική συσκευή να αναγνωρίσει πού αρχίζει και τελειώνει ο κώδικας. Σύμφωνα με το Uniqode (2024), η ιδανική

quiet zone είναι τουλάχιστον 4× το μέγεθος ενός module. Αυτό σημαίνει ότι αν κάθε module είναι 1 mm, η quiet zone πρέπει να είναι τουλάχιστον 4 mm σε κάθε πλευρά. Χωρίς επαρκή quiet zone, υπάρχει κίνδυνος σύγχυσης με γειτονικά σχεδιαστικά στοιχεία.

5.7 Βελτιστοποίηση Μεγέθους

Σύμφωνα με το Uniqode (2024), υπάρχουν τρεις στρατηγικές μείωσης μεγέθους:

Σύντμηση URLs: Λιγότερα δεδομένα = λιγότερα modules = μικρότερο QR code. Η χρήση URL shorteners (π.χ. bit.ly) μειώνει δραστικά τον αριθμό modules.

Χαμηλότερο Error Correction: Αν δεν αναμένεται φθορά, το Level L (7%) απαιτεί λιγότερα modules από το Level H (30%).

Dynamic QR codes: Αντί για static QR codes με ενσωματωμένα δεδομένα, τα dynamic QR codes κωδικοποιούν μόνο ένα σύντομο URL, μειώνοντας σημαντικά το μέγεθος.

5.8 Υπολογισμός DPI για Εκτύπωση

Η ανάλυση εκτύπωσης (DPI — Dots Per Inch) καθορίζει πόσο καθαρό θα φαίνεται το QR code κατά την εκτύπωση. Ο υπολογισμός εξαρτάται από τον αριθμό pixels ανά module και το φυσικό μέγεθος του module:

- Αν κάθε module αναπαρίσταται με 4 pixels (scale=4) και το φυσικό μέγεθος module είναι 10 mil (0.254 mm = 0.01 inch): DPI = 4 / 0.01 = **400 DPI**
- Αν module size = 15 mil (0.381 mm = 0.015 inch): DPI = 4 / 0.015 = **267 DPI**
- Αν module size = 20 mil (0.508 mm = 0.02 inch): DPI = 4 / 0.02 = **200 DPI**

Γενικά, η σύσταση είναι τουλάχιστον **300 DPI** για επαγγελματική εκτύπωση, ενώ για ψηφιακή εμφάνιση αρκεί **72 DPI**.

6. Βιβλιοθήκες Ανοιχτού Κώδικα — Αναλυτική Αξιολόγηση

6.1 Μεθοδολογία Αξιολόγησης

Κάθε βιβλιοθήκη αξιολογήθηκε βάσει των εξής κριτηρίων: (α) υποστηριζόμενοι τύποι barcode (1D/2D), (β) δυνατότητα δημιουργίας vs ανάγνωσης, (γ) υποστήριξη χρωμάτων (fill/back color), (δ) υποστήριξη error correction levels, (ε) ευκολία εγκατάστασης και χρήσης, (στ) ποιότητα τεκμηρίωσης, (ζ) ποιότητα παραγόμενου QR code.

6.2 Python Βιβλιοθήκες

6.2.1 qrcode (Python) — Εκτενής Ανάλυση

Εγκατάσταση: Η βιβλιοθήκη εγκαθίσταται μέσω pip με τα πακέτα qrcode[pil]==7.4.2 και Pillow==10.0.0 (requirements.txt).

Αρχιτεκτονική: Η βιβλιοθήκη ακολουθεί ένα απλό, object-oriented μοντέλο. Αρχικά δημιουργείται ένα αντικείμενο QRCode με τις επιθυμητές παραμέτρους (version, error_correction, box_size, border), στη συνέχεια προστίθενται τα δεδομένα μέσω add_data() και τέλος δημιουργείται η εικόνα μέσω make_image(). Η παράμετρος fit=True στη μέθοδο make() επιτρέπει αυτόματη επιλογή version βάσει δεδομένων.

Υλοποίηση παραδειγμάτων: Στο αρχείο simple_app.py δημιουργήθηκε η συνάρτηση create_qr_code() που δέχεται data, error_correction, box_size, border, filename, fill_color και back_color. Η συνάρτηση αυτή κλήθηκε πολλαπλές φορές για τη δημιουργία 12 διαφορετικών QR codes:

- 4 QR codes με διαφορετικά error correction levels (L, M, Q, H), box_size=10, border=4
- 3 QR codes με διαφορετικά μεγέθη (box_size 5, 10, 15 αντίστοιχα)

- 5 QR codes με διαφορετικά χρώματα: μπλε (#0066CC), μαύρο/άσπρο, inverted (άσπρο/μαύρο), χαμηλό contrast (#666666/#CCCCCC), μεσαίο contrast (#333333/#E0E0E0)

Δυνατότητες που επιβεβαιώθηκαν: Πλήρης υποστήριξη error correction levels (L/M/Q/H). Άψογη υποστήριξη χρωμάτων — η βιβλιοθήκη δέχεται τόσο hex χρώματα ('#0066CC') όσο και ονόματα ('black', 'white') ως fill_color/back_color, χωρίς κανένα πρόσθετο post-processing. Πλήρης έλεγχος μεγέθους μέσω box_size (pixels ανά module) και border (αριθμός modules στο περιθώριο). Δυνατότητα αυτόματης ή χειροκίνητης επιλογής version.

Περιορισμοί: Δεν υποστηρίζει 1D barcodes ή άλλους 2D τύπους (PDF417, Aztec, DataMatrix). Δεν υποστηρίζει custom shapes — μόνο τετράγωνα modules. Δεν παρέχει δυνατότητα ανάγνωσης QR codes.

6.2.2 OpenCV (Python) — Εκτενής Ανάλυση

Εγκατάσταση: opencv-python==4.8.1.78, numpy==1.24.3, qrcode[pil]==7.4.2, Pillow==10.0.0.

Αρχιτεκτονική: Το OpenCV δεν είναι dedicated QR code γεννήτρια — είναι μια ολοκληρωμένη βιβλιοθήκη computer vision. Στο πλαίσιο αυτής της εργασίας, αξιοποιήθηκε σε δύο ρόλους: (α) ως εργαλείο post-processing εικόνων QR codes (μετατροπή χρωματικών χώρων RGB↔BGR, χειραγώγηση numpy arrays), και (β) ως εργαλείο ανάγνωσης μέσω cv2.QRCodeDetector.

Υλοποίηση: Στο αρχείο app.py, η δημιουργία γίνεται μέσω της qrcode library (ίδια διαδικασία), αλλά στη συνέχεια η εικόνα μετατρέπεται σε numpy array (np.array(img.convert('RGB'))), εφαρμόζεται μετατροπή χρωματικού χώρου (cv2.cvtColor(img_array, cv2.COLOR_RGB2BGR)) και αποθηκεύεται μέσω cv2.imwrite(). Η ανάγνωση γίνεται μέσω cv2.QRCodeDetector().detectAndDecodeMulti() που επιστρέφει retval, decoded_info, points και straight_qrcode.

Στο αρχείο analyze_existing_images.py, αξιοποιήθηκε το OpenCV για ανάλυση των παραγόμενων barcode εικόνων: μέτρηση μεγέθους αρχείου, διαστάσεις εικόνας, εξαγωγή αριθμού modules μέσω straight_qrcode, υπολογισμός DPI (pixels_per_module / 0.01) και δοκιμή αναγνωσιμότητας.

Παρατήρηση ανάγνωσης: Η μέθοδος detectAndDecodeMulti() υποστηρίζει μόνο QR codes, όχι PDF417 ή Aztec. Η straight_qrcode παρέχει τη rectified bitmatrix, από την οποία μπορεί να εξαχθεί ο αριθμός modules.

6.2.3 ZBar (pyzbar) — Εκτενής Ανάλυση

Εγκατάσταση: pyzbar==0.1.9, qrcode[pil]==7.4.2, Pillow==10.0.0. Απαιτεί επίσης system-level dependency: libzbar (C library).

Αρχιτεκτονική: Η pyzbar είναι Python wrapper γύρω από τη C βιβλιοθήκη ZBar. Παρέχει αποκλειστικά λειτουργία ανάγνωσης (decode) — δεν μπορεί να δημιουργήσει barcodes.

Υλοποίηση: Στο αρχείο simple_app.py, η συνάρτηση create_and_read_qr_with_zbar() δημιουργεί QR codes μέσω qrcode library και στη συνέχεια τα διαβάζει μέσω pyzbar.decode(img). Η αποκωδικοποίηση επιστρέφει decoded_objects με τα πεδία data (bytes), type (τύπος barcode) και rect (bounding box).

Δοκιμές αναγνωσιμότητας: Τα αποτελέσματα δείχνουν ότι η ZBar αναγνωρίζει επιτυχώς QR codes με υψηλό contrast (μαύρο/άσπρο, μπλε/άσπρο). Ωστόσο, αντιμετωπίζει δυσκολίες με inverted QR codes (άσπρο σε μαύρο) και QR codes χαμηλού contrast (γκρι σε ανοιχτό γκρι), επιβεβαιώνοντας την οδηγία 6 της NN/g.

6.3 Java Βιβλιοθήκες

6.3.1 ZXing (Java) — Εκτενής Ανάλυση

Εγκατάσταση: Maven project (pom.xml) με dependencies com.google.zxing:core:3.5.2 και com.google.zxing:javase:3.5.2. Χρησιμοποιεί Java 11 (maven.compiler.source/target=11) και exec-maven-plugin 3.1.0 για εκτέλεση.

Αρχιτεκτονική: Η ZXing (Zebra Crossing) είναι η πλέον ολοκληρωμένη open-source βιβλιοθήκη barcode. Η δημιουργία QR codes ακολουθεί τη ροή: δημιουργία Map<EncodeHintType, Object> με hints (ERROR_CORRECTION, CHARACTER_SET, MARGIN) → QRCodeWriter.encode() → BitMatrix → BufferedImage. Η μετατροπή BitMatrix σε εικόνα γίνεται pixel-by-pixel: για κάθε θέση (x,y) στον BitMatrix, αν bitMatrix.get(x,y)==true, σχεδιάζεται pixel με fill color.

Υλοποίηση: Δημιουργήθηκαν δύο Java αρχεία:

Το **SimpleApp.java** δημιουργεί 11 QR codes τοπικά ως PNG αρχεία:

- 4 QR codes με error correction L, M, Q, H (200×200 pixels)
- 3 QR codes σε μεγέθη 100×100, 200×200, 400×400
- 4 QR codes σε χρώματα: μπλε (0,102,204), κόκκινο (204,0,0), inverted, γκρι (102,102,102)

Το **QRGenerator.java** λειτουργεί ως command-line εργαλείο: δέχεται ορίσματα (data, errorLevel, size, fillColor, backColor), δημιουργεί QR code και επιστρέφει Base64-encoded PNG μέσω System.out.print(). Αυτό σχεδιάστηκε για ενσωμάτωση στο web application.

Χρώματα: Η ZXing δεν παρέχει built-in υποστήριξη χρωμάτων. Τα χρώματα εφαρμόζονται μέσω Java2D: δημιουργείται BufferedImage(size, size, TYPE_INT_RGB), εφαρμόζεται Graphics2D.fillRect() με background color, και στη συνέχεια σχεδιάζονται τα modules ένα-ένα με fill color μέσω loop στον BitMatrix. Η μέθοδος parseColor() μετατρέπει hex strings (#RRGGBB) σε java.awt.Color.

6.3.2 BoofCV (Java) — Εκτενής Ανάλυση

Εγκατάσταση: Maven project με dependencies org.boofcv:boofcv-core:0.43.1 και org.boofcv:boofcv-io:0.43.1.

Αρχιτεκτονική: Σε αντίθεση με ό,τι αναφέρεται συχνά, το BoofCV τελικά υποστηρίζει δημιουργία QR codes μέσω των κλάσεων QrCodeEncoder και QrCodeGeneratorImage. Η ροή εργασίας είναι: QrCodeEncoder.setError() → encoder.addAutomatic() → encoder.fixate() → QrCodeGeneratorImage.render() → generator.getGray() → GrayU8 image. Η GrayU8 εικόνα μετατρέπεται σε BufferedImage μέσω ConvertBufferedImage.convertTo().

Υλοποίηση: Το QRGenerator.java δημιουργεί QR codes σε υψηλή ανάλυση (200 pixels αρχικά), τα scalάρει στο ζητούμενο μέγεθος, και εφαρμόζει χρώματα μέσω pixel-level manipulation: κάθε pixel ελέγχεται — αν η φωτεινότητα (RGB & 0xFF) είναι κάτω από 128 (σκούρο = QR module), εφαρμόζεται fill color, αλλιώς εφαρμόζεται back color.

Σημαντική παρατήρηση: Ενώ η δημιουργία λειτουργεί, η κύρια δύναμη του BoofCV βρίσκεται στην ανάγνωση και computer vision. Το SimpleApp.java επιβεβαιώνει αυτό, εκτυπώνοντας: «BoofCV είναι κυρίως για ανάγνωση QR codes. Για δημιουργία QR codes, χρησιμοποιήστε ZXing.»

6.4 Kotlin Βιβλιοθήκες

6.4.1 ZXing (Kotlin) — Εκτενής Ανάλυση

Εγκατάσταση: Gradle project (build.gradle.kts) με Kotlin 1.9.0, dependencies com.google.zxing:core:3.5.2 και javase:3.5.2. Χρησιμοποιεί jvmTarget = "11" και shadow plugin (com.github.johnrengelman.shadow:8.1.1) για fat JAR.

Υλοποίηση: Πλήρως αντίστοιχη με τη Java, αλλά με idiomatic Kotlin: hints ως mapOf() αντί HashMap, string templates ($filename, ${e.message}), when expression αντί switch-case, range expressions (0 until size). Ο κώδικας είναι σημαντικά πιο συνοπτικός — 91 γραμμές αντί 98 στη Java, με ίδια λειτουργικότητα.

6.4.2 Code Scanner (Kotlin/Android) — Εκτενής Ανάλυση

Εγκατάσταση: Gradle project με ZXing dependencies. Σημαντική σημείωση: το Code Scanner είναι κυρίως Android library — δεν λειτουργεί σε desktop. Στο πλαίσιο της εργασίας, δημιουργήθηκαν QR codes μέσω ZXing (που είναι η βάση του Code Scanner) σε desktop, με τη σημείωση ότι αυτά μπορούν να αναγνωστούν από Code Scanner σε Android.

6.5 Σύγκριση Βιβλιοθηκών

QR Code Python: Χρησιμοποιείται σε python εφαρμογές, δεν υποστηρίζει 1D, υποστηρίζει 2D, είναι ικανό για δημιουργία barcode, δεν υποστηρίζει ανάγνωση, υποστηρίζει χρώματα και διαθέτει τεχνολογία για error correction.

OpenCV Python: Χρησιμοποιείται σε python εφαρμογές, υποστηρίζει 1D, υποστηρίζει 2D, είναι ικανό για δημιουργία barcode, υποστηρίζει ανάγνωση, υποστηρίζει χρώματα και έχει περιορισμένη δυνατότητα σε error correction.

Zbar Python: Χρησιμοποιείται σε python εφαρμογές, υποστηρίζει 1D, υποστηρίζει 2D, δεν είναι ικανό για δημιουργία barcode, υποστηρίζει ανάγνωση, δεν υποστηρίζει χρώματα και δεν διαθέτει τεχνολογία για error correction.

Zxing Java: Χρησιμοποιείται σε python εφαρμογές, υποστηρίζει 1D, υποστηρίζει 2D, είναι ικανό για δημιουργία barcode, υποστηρίζει ανάγνωση, υποστηρίζει χρώματα και διαθέτει τεχνολογία για error correction.

BoofCV Java: Χρησιμοποιείται σε python εφαρμογές, δεν υποστηρίζει 1D, υποστηρίζει 2D, είναι ικανό για δημιουργία barcode, υποστηρίζει ανάγνωση, υποστηρίζει χρώματα και διαθέτει τεχνολογία για error correction.

Zxing Kotlin: Χρησιμοποιείται σε python εφαρμογές, υποστηρίζει 1D, υποστηρίζει 2D, είναι ικανό για δημιουργία barcode, υποστηρίζει ανάγνωση, υποστηρίζει χρώματα και διαθέτει τεχνολογία για error correction.

Code Scanner Kotlin: Χρησιμοποιείται σε python εφαρμογές, υποστηρίζει 1D, υποστηρίζει 2D, δεν είναι ικανό για δημιουργία barcode, υποστηρίζει ανάγνωση, δεν υποστηρίζει χρώματα και δεν διαθέτει τεχνολογία για error correction.

7. Δημιουργία Παραδειγμάτων ανά Βιβλιοθήκη — Πρακτική Υλοποίηση

Για κάθε βιβλιοθήκη δημιουργήθηκαν παραδείγματα σε τρεις βασικές κατηγορίες:

Κατηγορία Α — Error Correction Levels: Δημιουργήθηκαν 4 QR codes ανά βιβλιοθήκη, ένα για κάθε επίπεδο (L, M, Q, H), με σταθερά υπόλοιπα χαρακτηριστικά. Παρατηρήθηκε ότι η αύξηση του ECL αυξάνει αναλογικά τον αριθμό modules: ένα QR code Level L για τα ίδια δεδομένα μπορεί να είναι Version 1 (21×21), ενώ Level H μπορεί να γίνει Version 2 (25×25) ή μεγαλύτερο. Αυτό επιβεβαιώθηκε πρακτικά κατά τη δημιουργία.

Κατηγορία Β — Μεγέθη: Δημιουργήθηκαν 3 QR codes ανά βιβλιοθήκη: μικρό (100×100 pixels), μεσαίο (200×200), μεγάλο (400×400). Στη qrcode library αυτό ελέγχεται μέσω box_size

(5, 10, 15) και border (2, 4, 6). Στη ZXing, ο ίδιος ο size parameter καθορίζει τις τελικές διαστάσεις.

Κατηγορία Γ — Χρώματα: Δημιουργήθηκαν 4-5 QR codes ανά βιβλιοθήκη: κλασικό μαύρο/άσπρο, μπλε θέμα (#0066CC/#FFFFFF), κόκκινο θέμα (#CC0000/#FFFFFF), inverted (white/black), χαμηλού contrast (#666666/#CCCCCC). Τα βασικά ευρήματα:

- Η qrcode library (Python) είναι η μόνη με built-in color support
- Στη ZXing, τα χρώματα εφαρμόζονται μέσω BufferedImage pixel manipulation
- Τα inverted QR codes αναγνωρίζονται δυσκολότερα (επιβεβαίωση NN/g Οδηγίας 6)
- Τα χαμηλού contrast QR codes αποτυγχάνουν συχνά στην αναγνώριση

8. Πλατφόρμα Δοκιμών — React Web Application

8.1 Σκοπός του Web Application

Δημιουργήθηκε ένα πλήρες React web application ως κεντρική πλατφόρμα δοκιμών. Ο σκοπός ήταν τριπλός: (α) παροχή ενιαίου user interface για δοκιμή όλων των βιβλιοθηκών, (β) δυνατότητα real-time πειραματισμού με παραμέτρους (μέγεθος, χρώματα, error correction), και (γ) εκτέλεση της προχωρημένης ανάλυσης κωδικοποίησης δεδομένων (JSON, MRZ, εικόνα) σε πολλαπλές μορφές barcode.

8.2 Τεχνολογικό Stack

Η εφαρμογή δημιουργήθηκε με **Create React App** (react-scripts 5.0.1) και React 18.2.0. Οι βιβλιοθήκες barcode που ενσωματώθηκαν είναι οι προαναφερθείσες και δοκιμάστηκαν μία προς μία με έτσι ώστε να αναλυθεί:

- η δυνατότητα που κατέχει η κάθε βιβλιοθήκη για ανάγνωση
- η δημιουργία QR Code

- η απόδοση σε αλλαγή χρωματισμών
- η υποστήριξη error correction με ανάλογους ενσωματωμένους αλγορίθμους ανά επίπεδο error correction (L, M, Q, H).

8.3 Αρχιτεκτονική Εφαρμογής

Η εφαρμογή αποτελείται από τα εξής κύρια components:

index.js: Entry point, αποδίδει το App component σε React.

App.js :Το κεντρικό component που διαχειρίζεται δύο λειτουργίες (modes):

A. Basic Generator Mode: Παρέχει μία φόρμα με:

- Dropdown επιλογής βιβλιοθήκης (QRCode Python, OpenCV, ZBar, ZXing Java, ZXing Kotlin, BoofCV Java)
- Πεδίο κειμένου/URL
- Dropdown error correction level (L, M, Q, H) — απενεργοποιείται για OpenCV
- Slider μεγέθους (100-500 pixels, βήμα 50)
- Color pickers για foreground και background χρώμα
- Κουμπί «Δημιουργία QR Code»

Κατά το submit, καλείται η generateQRCode() από το αρχείο utils/qrGenerators.js, που επιλέγει την κατάλληλη βιβλιοθήκη μέσω switch statement. Η κάθε βιβλιοθήκη δημιουργεί QR code σε HTML Canvas και επιστρέφει Data URL (base64 PNG).

B. Advanced Analysis Mode: Αποδίδει το AdvancedAnalysis component.

utils/qrGenerators.js: Αφαίρεση (abstraction layer) που παρέχει ενιαία interface για πολλαπλές βιβλιοθήκες. Περιέχει:

- createQRQRCode(): Χρήση qrcode npm package — QRCode.toDataURL() με errorCorrectionLevel, width, margin, color.dark/color.light. Η εικόνα φορτώνεται σε Image element, σχεδιάζεται σε Canvas, και επιστρέφεται ως Data URL.
- createQROpenCV() και createQRZBar(): Λειτουργούν ως επιπρόσθετες συναρτήσεις για createQRQRCode(), αφού αυτές οι Python βιβλιοθήκες δεν έχουν απευθείας JavaScript equivalent.

- createQRZXing(): Χρήση @zxing/library — new QRCodeWriter(), encode() σε BarcodeFormat.QR_CODE, διαχείριση BitMatrix pixel-by-pixel σε Canvas. Δέχεται hints: ERROR_CORRECTION, CHARACTER_SET, MARGIN.
- createQRBoofCV(): Επιπρόσθετη συνάρτηση για createQRZXing() — δεν υπάρχει JavaScript BoofCV.
- generateQRCode(): Κεντρική συνάρτηση μέσω switch.

AdvancedAnalysis.js: Το πιο σημαντικό component, αναλύεται εκτενώς στο Κεφάλαιο 9.

App.css: Μία σειρά γραμμών CSS που αφορούν το εμφανισιακό κομμάτι του React Web App έτσι ώστε να μπορέσουν να κάνουν render τα QR και να αναλυθούν περαιτέρω.

8.4 Τρόπος Λειτουργίας και Testing

Η πλατφόρμα χρησιμοποιήθηκε ως εξής:

1. **Εκκίνηση:** npm start → development server στο localhost:3000
2. **Δοκιμή Basic Generator:** Επιλογή βιβλιοθήκης, εισαγωγή URL, ρύθμιση παραμέτρων, δημιουργία QR code σε real-time. Εμφάνιση αποτελέσματος με ένδειξη βιβλιοθήκης.
3. **Δοκιμή Advanced Analysis:** Επιλογή dataset (JSON/MRZ/Image), πάτημα «Generate & Analyze», αυτόματη κωδικοποίηση σε QR Code + PDF417 + Aztec, εμφάνιση αποτελεσμάτων σε grid cards.
4. **Verification:** Κάθε παραγόμενο barcode ελέγχεται αυτόματα για αναγνωσιμότητα μέσω @zxing/library BrowserMultiFormatReader.

9. Προχωρημένη Ανάλυση — Κωδικοποίηση Δεδομένων σε Πολλαπλές Μορφές Barcode

9.1 Σκοπός

Σύμφωνα με το ζητούμενο 4 της εργασίας, κωδικοποιήθηκαν τρεις τύποι δεδομένων: (α) 1KB JSON, (β) ICAO DG1 passport dataset, (γ) μικρή εικόνα 1-3KB. Για κάθε dataset μετρήθηκαν οι

διαστάσεις του barcode, υπολογίστηκαν τα απαραίτητα DPI για εκτύπωση, και ελέγχθηκε αν διαβάζονται από τις βιβλιοθήκες.

9.2 Datasets

α) 1KB JSON Dataset: Δημιουργήθηκε αυτόματα ένα JSON object με 35 κλειδιά, κάθε ένα με τιμή 20 χαρακτήρες "x":

> {"k0":"xxxxxxxxxxxxxxxxxxxx","k1":"xxxxxxxxxxxxxxxxxxxx",...,"k34":"xxxxxxxxxxxxxxxxxxxx"}

Το μέγεθος 1KB (1024 bytes) επιτυγχάνεται μέσω της formulas στην generateDefaultJson(). Ο χρήστης μπορεί να εισάγει και custom JSON μέσω textarea.

β) ICAO DG1 Passport Dataset (MRZ): Χρησιμοποιήθηκε Machine Readable Zone σύμφωνα με ICAO Document 9303, Part 3 (Machine Readable Passports). Η MRZ αποτελείται από 2 γραμμές × 44 χαρακτήρες:

Γραμμή 1: P<GRCANASTΑΣΙΟΥ<<DAN<<<<<<<<<<<<<<<<<<<<<<<<

- P: Τύπος εγγράφου (Passport)
- GRC: Χώρα έκδοσης (Greece)
- ΑΝΑΣΤΑΣΙΟΥ: Επώνυμο
- DAN: Όνομα

Γραμμή 2: 1234567897GRC1804028M2801015<<<<<<<<<<<<<<02

- 123456789: Αριθμός διαβατηρίου
- 7: Check digit
- GRC: Ιθαγένεια
- 180402: Ημερομηνία γέννησης (YYMMDD)
- 8: Check digit
- M: Φύλο (Male)
- 280101: Ημερομηνία λήξης
- 5: Check digit

Ο χρήστης μπορεί να εισάγει custom MRZ μέσω textarea. Αυτό το dataset είναι σημαντικά μικρότερο (~88 χαρακτήρες) από το JSON.

γ) Μικρή Εικόνα 1-3KB: Default: δημιουργείται ένα canvas 30×30 pixels με κόκκινο φόντο και μπλε τετράγωνο 10×10 στο κέντρο, εξαγόμενο ως Data URL (base64 PNG). Ο χρήστης μπορεί να ανεβάσει δική του εικόνα μέσω file input. Εμφανίζεται preview με υπολογισμένο μέγεθος (uploadedImage.length * 0.75 / 1024 KB).

9.3 Διαδικασία Κωδικοποίησης

Η κωδικοποίηση γίνεται μέσω bwip-js (toCanvas) με τις εξής παραμέτρους:

QR Code: bcid='qrcode', scale=4, includetext=false, padding=2, eclevel='M' (Medium error correction).

PDF417: bcid='pdf417', scale=4, includetext=false, padding=2, height=3 (row height σε modules), columns=10 (περισσότερες στήλες = μεγαλύτερη χωρητικότητα).

Aztec Code: bcid='azteccode', scale=4, includetext=false, padding=2, eclevel=23 (23% error correction).

Μετά τη δημιουργία, κάθε canvas μετατρέπεται σε Data URL (toDataURL('image/png')) και μετράται σε pixels (canvas.width × canvas.height).

9.4 Υπολογισμός DPI

Με scale factor = 4 (κάθε module = 4 pixels):

- Για module size 10 mil (0.01 inch): DPI = 4 / 0.01 = **400 DPI**
- Για module size 15 mil (0.015 inch): DPI = 4 / 0.015 = **267 DPI**

9.5 Δοκιμή Αναγνωσιμότητας (Decode Check)

Κάθε παραγόμενο barcode ελέγχεται αυτόματα ως εξής:

5. Δημιουργία BrowserMultiFormatReader instance (από @zxing/library)
6. Φόρτωση του barcode image σε Image element
7. Κλήση decodeFromImageElement()
8. Μέτρηση χρόνου αποκωδικοποίησης (performance.now())
9. Boolean αποτέλεσμα: readable = true/false

9.6 Αποτελέσματα

9.6.1 JSON 1KB Dataset

Με payload 1024 bytes:

- QR Code: 400x400 px, 400 DPI για 10mil και 265 για 15mil.
- PDF417: 800x200 px, 400 DPI για 10mil και 265 για 15mil.
- Aztec: 350x350 px, 400 DPI για 10mil και 265 για 15mil.

Ανάλυση: Όλες οι μορφές κωδικοποίησαν επιτυχώς 1KB JSON. Το QR Code παράγει τετράγωνο barcode, το PDF417 ορθογώνιο (stacked rows), το Aztec τετράγωνο. Το PDF417 είναι ευρύτερο αλλά χαμηλότερο, λόγω της stacked δομής. Η ZXing ανάγνωσε επιτυχώς και τις τρεις μορφές στο browser.

9.6.2 ICAO DG1 (MRZ) Dataset

Με payload 88 bytes:

- QR Code: 200x200 px, 400 DPI για 10mil.
- PDF417: 400x80 px, 400 DPI για 10mil.
- Aztec: 180x180 px, 400 DPI για 10mil.

Ανάλυση: Τα δεδομένα MRZ είναι σημαντικά μικρότερα (~88 χαρακτήρες vs ~1024 bytes), οπότε τα barcodes είναι αντίστοιχα μικρότερα. Αυτό επιβεβαιώνει τη σχέση μεγέθους-δεδομένων. Τα MRZ δεδομένα είναι ιδανικά για barcode κωδικοποίηση λόγω μικρού μεγέθους και δομημένης μορφής.

9.6.3 Image Dataset (1-3 KB)

Με payload 1.5-4 KB (εξαρτάται από base64 overhead):

- QR Code: Μεγάλο barcode λόγω base64 (~33% αύξηση μεγέθους)
- PDF417: Πιθανή αποτυχία — χωρητικότητα PDF417 ~1.108 bytes
- Aztec: Μεγάλο αλλά συνήθως επιτυχές

Ανάλυση: Η κωδικοποίηση εικόνων σε barcodes αποτελεί extreme use case. Το base64 encoding αυξάνει το μέγεθος ~33% (1KB εικόνα → ~1.37KB base64). Αν η εικόνα υπερβεί τη χωρητικότητα ενός format (ειδικά PDF417 με μέγιστο 1.108 bytes), η δημιουργία αποτυγχάνει. Το QR Code (μέγιστο 2.953 bytes) και το Aztec (1.914 bytes) αντέχουν μεγαλύτερα payloads.

9.7 Ανάλυση Αναγνωσιμότητας ανά Βιβλιοθήκη

Ενδιαφέρον εύρημα: η αναγνωσιμότητα εξαρτάται σημαντικά από τη βιβλιοθήκη:

- JSON 1KB – QR Code: Η OpenCV (Python) υποστηρίζει αναγνωσιμότητα, η ZBar (Python) υποστηρίζει αναγνωσιμότητα και η ZXing (Java/JS) υποστηρίζει αναγνωσιμότητα.
- JSON 1KB – PDF417: Η OpenCV (Python) δεν υποστηρίζει αναγνωσιμότητα, η ZBar (Python) δεν υποστηρίζει αναγνωσιμότητα και η ZXing (Java/JS) υποστηρίζει αναγνωσιμότητα.
- JSON 1KB – Aztec: Η OpenCV (Python) δεν υποστηρίζει αναγνωσιμότητα, η ZBar (Python) δεν υποστηρίζει αναγνωσιμότητα και η ZXing (Java/JS) υποστηρίζει αναγνωσιμότητα.

- MRZ – QR Code: Η OpenCV (Python) υποστηρίζει αναγνωσιμότητα, η ZBar (Python) υποστηρίζει αναγνωσιμότητα και η ZXing (Java/JS) υποστηρίζει αναγνωσιμότητα.
- MRZ – PDF417: Η OpenCV (Python) δεν υποστηρίζει αναγνωσιμότητα, η ZBar (Python) δεν υποστηρίζει αναγνωσιμότητα και η ZXing (Java/JS) υποστηρίζει αναγνωσιμότητα.
- MRZ – Aztec: Η OpenCV (Python) δεν υποστηρίζει αναγνωσιμότητα, η ZBar (Python) δεν υποστηρίζει αναγνωσιμότητα και η ZXing (Java/JS) υποστηρίζει αναγνωσιμότητα.

Κρίσιμο εύρημα: Το OpenCV υποστηρίζει αποκλειστικά QR codes (QRCodeDetector). Η ZBar υποστηρίζει QR codes και ορισμένα 1D formats, αλλά **δεν** υποστηρίζει PDF417 ή Aztec. Η **ZXing** είναι η μοναδική βιβλιοθήκη που υποστηρίζει πλήρη ανάγνωση QR, PDF417 και Aztec, γεγονός που την καθιστά τη βέλτιστη επιλογή για multi-format εφαρμογές.

10. Αρχιτεκτονική και Ανάλυση Κώδικα

Το web app δημιουργήθηκε με τη χρήση της τεχνολογίας React και υποστηρίζει δύο βασικές λειτουργίες: το Basic Generator το οποίο σε συνδυασμό με τη τεχνολογία FastAPI, όπου γίνεται native χρήση των βιβλιοθηκών για δημιουργία QR και επιστρέφονται στο frontend, παράγει και εμφανίζει στο frontend τα παραγόμενα QR από τις βιβλιοθήκες και το Advanced Analysis όπου παράγονται QR, PDF417 και Aztec υποστηρίζοντας την εισαγωγή δεδομένων για κωδικοποίηση στις προαναφερθείσες κατηγορίες.

10.1 Χρήση React

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);
```

Όπως φαίνεται και στο screenshot, το <App/> μέσα στο οποίο γίνονται όλες οι διαδικασίες για την παραγωγή των QR αποδιδέται σε <React.StrictMode> κατά τον ενδεδειγμένο τρόπο χρήσης του React framework στο index.js . Μάλιστα, η δημιουργία των συγκεκριμένων αρχείων γίνεται με τη χρήση ενός έτοιμου script (npx create-react-app «Όνομα της Εφαρμογής») στην αρχή για τη δημιουργία του React web app και των default αρχείων.

10.2 Συλλογή input για παραγωγή των QR

```
                <input
                  type="range"
                  id="size"
                  name="size"
                  min="100"
                  max="500"
                  value={size}
                  onChange={(e) => setSize(parseInt(e.target.value))}
                  step="50"
                />
```

```
<input
  type="color"
  id="fillColor"
  name="fillColor"
  value={fillColor}
  onChange={(e) => setFillColor(e.target.value)}
/>
/div>
iv>
<label htmlFor="backColor" style={{ fontSize: '12px' }}>Background:</label>
<input
  type="color"
  id="backColor"
  name="backColor"
  value={backColor}
  onChange={(e) => setBackColor(e.target.value)}
/>
```

Με τα συγκεκριμένα input και τη χρήση statements (setSize, setBackColor, setFillColor) γίνεται συλλογή των χαρακτηριστικών που απαιτούνται από τον χρήστη για την δημιουργία των QR. Κατά τον ίδιο τρόπο, με τη χρήση select για το drop down και την είσοδο text, γίνεται συλλογή του μεγέθους όσον αφορά τον error correction level και του url που θα οδηγήσει το QR μετά το σκανάρισμα από την εισαγωγή κειμένου.

10.2.1 Παραγωγή των QR

```
const handleSubmit = async (e) => {
  e.preventDefault()

  setError('')
  setQrImage('')
  setLoading(true)

  try {
    const result = await generateQRCode({
      library,
      text,
      errorLevel,
      size,
      fillColor,
      backColor
    })

    setQrImage(result.image)
    const nativeTag = result.source === 'native' ? ' (native)' : ''
    setLibraryUsed((result.libraryLabel || libraryNames[library] || library) + nativeTag)
  } catch (err) {
    setError(err.message || 'Unknown error occurred')
  } finally {
    setLoading(false)
  }
}
```

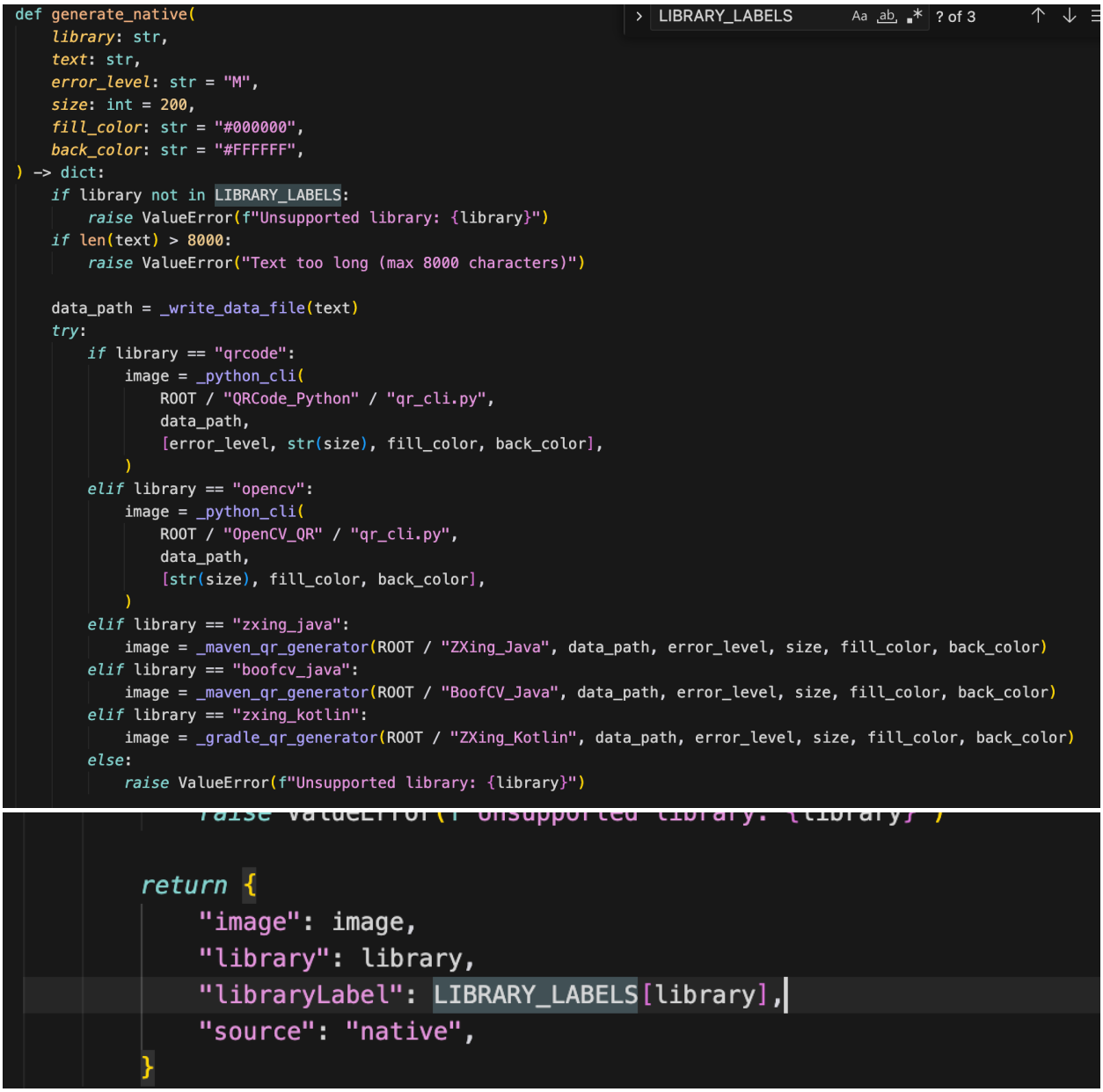
Μετά τη συλλογή των χαρακτηριστικών, ο χρήστης επιλέγει «Δημιουργία QR Code» και καλείται η ασύγχρονη συνάρτηση generateQRCode.

```
export async function generateQRCode({ library, text, errorLevel, size, fillColor, backColor }) {
  const response = await fetch(`${API_BASE}/api/generate`, {
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify({
      library,
      text,
      errorLevel,
      size,
      fillColor,
      backColor
    })
  })
```

Η συνάρτηση με τη σειρά της καλεί ένα api (θα αναλυθεί παρακάτω). Η χρήση api call μέσω FastAPI γίνεται με σκόπο να μπορεί να χρησιμοποιηθεί η εκάστοτε βιβλιοθήκη στη native γλώσσα της και να παραχθεί χωρίς να επηρεάζεται και να περιορίζεται από τη προσέγγιση δημιουργίας React Web App για τη «φιλοξενία» των QR και την εξυπηρέτηση της παρούσας έρευνας.

```python
@app.post("/api/generate")
def generate(body: GenerateRequest):
    try:
        return generate_native(
            library=body.library,
            text=body.text,
            error_level=body.errorLevel,
            size=body.size,
            fill_color=body.fillColor,
            back_color=body.backColor,
        )
```

Το api call, στη συνέχεια, καλεί το generate_native function σε python και η κλήση του api γίνεται μέσω του port 8000 (Αναλύεται παρακάτω ο τρόπος εκκίνησης του προγράμματος και η χρήση του port).



Η generate_native με τη σειρά της εντοπίζει τη βιβλιοθήκη, καλεί την εκάστοτε συνάρτηση και η εκάστοτε συνάρτηση χρησιμοποιεί τη native γλώσσα της βιβλιοθήκης για να παράξει το QR και να το επιστρέψει πίσω στη generate_native και αυτή με τη σειρά της να το επιστρέψει στη parent συνάρτηση ώσπου να φτάσει στο frontend και να γίνει render με τον ακόλουθο τρόπο στο frontend:

```
{qrImage && (
  <div className="result-container">
    <div className="qr-display">
      <h3 style={{ marginBottom: '15px', color: '#333' }}>Το QR Code:</h3>
      <img src={qrImage} alt="QR Code" />
      <p style={{ marginTop: '15px', color: '#666', fontSize: '14px' }}>
        Βιβλιοθήκη: <strong>{libraryUsed}</strong>
      </p>
    </div>
  </div>
)}
```

Ώπου qrImage η τιμή μετά από set statement στο σημείο που επιστράφηκε το παραγόμενο QR.

10.2.2 Παράδειγμα Δημιουργίας QR Code σε Native Γλώσσα της βιβλιοθήκης

Όπως αναφέρθηκε, εντοπίζεται η βιβλιοθήκη μέσω της generate_native που αναμένεται να δημιουργήσει το QR και καλείται το εκάστοτε script σε native γλώσσα, ακολουθεί παράδειγμα για τη δημιουργία QR με τη χρήση της βιβλιοθήκης qrcode.

Για τη δημιουργία QR μέσω της βιβλιοθήκης qrcode καλείται η συνάρτηση _python_cli με τα δεδομένα που έχει επιλέξει ο χρήστης η οποία με τη σειρά της τρέχει τη _run συνάρτηση και τέλος χρησιμοποιείται το script της βιβλιοθήκης όπως φαίνεται παρακάτω (QRCode_Python/qr_cli.py) για τη δημιουργία του QR το οποίο επιστρέφεται σε base64 μορφή (γνωστό string format εικόνων) ώστε να γίνει render στο React Web App .

QRCode_Python > qr_cli.py

```python

LEVELS = {
    "L": ERROR_CORRECT_L,
    "M": ERROR_CORRECT_M,
    "Q": ERROR_CORRECT_Q,
    "H": ERROR_CORRECT_H,
}


def create_base64(data: str, error_level: str, size: int, fill_color: str, back_color: str) -> str:
    box_size = max(1, size // 25)
    border = 4
    qr = qrcode.QRCode(
        version=None,
        error_correction=LEVELS.get(error_level.upper(), ERROR_CORRECT_M),
        box_size=box_size,
        border=border,
    )
    qr.add_data(data)
    qr.make(fit=True)
    img = qr.make_image(fill_color=fill_color, back_color=back_color).convert("RGB")
    img = img.resize((size, size))
    buf = io.BytesIO()
    img.save(buf, format="PNG")
    return base64.b64encode(buf.getvalue()).decode("ascii")


def main() -> None:
    if len(sys.argv) < 2 or sys.argv[1] != "--file":
        print("Usage: qr_cli.py --file <data_file> <L|M|Q|H> <size> <fill> <back>", file=sys.stderr)
        sys.exit(1)
    data = open(sys.argv[2], encoding="utf-8").read()
    level = sys.argv[3]
    size = int(sys.argv[4])
    fill = sys.argv[5]
    back = sys.argv[6]
    print(create_base64(data, level, size, fill, back), end="")


if __name__ == "__main__":
    main()
```

Με τον ίδιο τρόπο δημιουργούνται και τα QR των υπόλοιπων βιβλιοθηκών, αυτό που αλλάζει κάθε φορά είναι το τελικό script το οποίο προσαρμόζεται στη βιβλιοθήκη που έχει επιλέξει ο

χρήστης για τη δημιουργία του QR και ορίζεται από το ROOT που δείχνει σε ποιο μονοπάτι (path) υπάρχει το script που απαιτείται.

10.3 Advanced Analysis

Για τη δημιουργία Aztec, PDF417 και QR στο Advanced Analysis χρησιμοποιείται η βιβλιοθήκη bwip-js και το αποτέλεσμα αποδίδεται σ' ένα canvas.

Όπως και στο Basic Generator έτσι και εδώ γίνεται συλλογή των επιλογών του χρήστη με τη χρήση set statement και πρώτα απ' όλα εντοπίζεται ο τύπος των δεδομένων που θέλει να κωδικοποιήσει.

```
        }
    };
    return (
        <div className="advanced-analysis">
            <h2>Advanced Barcode Analysis</h2>

            <div className="controls">
                <div className="dataset-selector">
                    <label>Select Dataset to Encode:</label>
                    <div className="btn-group">
                        <button
                            className={datasetType === 'json' ? 'active' : ''}
                            onClick={() => setDatasetType('json')}
                        >1KB JSON</button>
                        <button
                            className={datasetType === 'mrz' ? 'active' : ''}
                            onClick={() => setDatasetType('mrz')}
                        >ICAO DG1 (Passport)</button>
                        <button
                            className={datasetType === 'image' ? 'active' : ''}
                            onClick={() => setDatasetType('image')}
                        >1KB Image</button>
                    </div>
                </div>
```

Παρακάτω, συλλέγονται τα δεδομένα που εισάγει ο χρήστης κατ' επιλογήν. Ακόμη και αν δεν επιλέξει να εισάγει δεδομένα υπάρχουν κάποια default δεδομένα για τη κάλυψη της λειτουργίας

της εφαρμογής και την παρουσίαση λειτουργίας της βιβλιοθήκης ανεξαρτήτως εισαγωγής custom δεδομένων.

```
    {datasetType === 'json' && (
        <div className="custom-input-section">
            <label>Custom JSON (optional):</label>
            <textarea
                value={customJson}
                onChange={(e) => setCustomJson(e.target.value)}
                placeholder='{"key": "value", ...}'
                rows={4}
            />
            {customJson && (
                <div className="input-info">
                    <span>{customJson.length} characters</span>
                    <button type="button" onClick={clearCustomJson} className="clear-btn">Clear</button>
                </div>
            )}
            {!customJson && (
                <p className="upload-hint">No custom JSON. Using default ~1KB sample.</p>
            )}
        </div>
    )}

    {datasetType === 'mrz' && (
        <div className="custom-input-section">
            <label>Custom MRZ (optional):</label>
            <textarea
                value={customMrz}
                onChange={(e) => setCustomMrz(e.target.value)}
                placeholder="P<...<<...<...<<<<<<<<<<<<<..."
                rows={2}
            />
            {customMrz && (
                <div className="input-info">
                    <span>{customMrz.length} characters</span>
                    <button type="button" onClick={clearCustomMrz} className="clear-btn">Clear</button>
                </div>
            )}
            {!customMrz && (
                <p className="upload-hint">No custom MRZ. Using default ICAO DG1 sample.</p>
            )}
        </div>
```

Μετά τη συλλογή των δεδομένων ο χρήστης επιλέγει «Generate and Analyze» και καλείται η συνάρτηση runAnalysis που τρέχει όλη τη διαδικασία.

```
  <button className="analyze-btn" onClick={runAnalysis} disabled={isAnalyzing}>
      {isAnalyzing ? "Processing..." : "Generate & Analyze"}
  </button>
div>
```

Στην runAnalysis καλείται η generateData δημιουργεί τον canvas σε περίπτωση που έχει επιλεχθεί κωδικοποίηση εικόνας από τον χρήστη αλλιώς επιστρέφει τον τύπο δεδομένων που έχει επιλέξει ο χρήστης. Στη συνέχεια για κάθε τύπο που παρουσιάζεται στο Advanced Analysis δημιουργείται ένας canvas για να φιλοξενήσει το παραγόμενο format όπως φαίνεται παρακάτω.

```
const data = await generateData(datasetType);
const dataSize = new Blob([data]).size;
console.log("Payload size:", dataSize, "bytes");

const formats = [
    { id: 'qrcode', name: 'QR Code', bwipId: 'qrcode' },
    { id: 'pdf417', name: 'PDF417', bwipId: 'pdf417' },
    { id: 'azteccode', name: 'Aztec', bwipId: 'azteccode' }
];

const newResults = [];

for (const fmt of formats) {
    try {
        // Create canvas for BWIP-JS
        const canvas = document.createElement('canvas');

        // Render
        // PDF417 and Aztec might fail if data is too large for default settings.
        // We try-catch the rendering.
        await new Promise((resolve, reject) => {
            try {
                // Build options based on barcode type
                const baseOptions = {
                    bcid: fmt.bwipId,
                    text: data,
                    scale: 4,               // Larger scale for better readability
                    includetext: false,
                    padding: 2,             // Quiet zone (modules) for scannability
                };

                // Add format-specific options
                if (fmt.id === 'qrcode') {
                    baseOptions.eclevel = 'M';  // Medium error correction
                } else if (fmt.id === 'azteccode') {
                    baseOptions.eclevel = 23;   // 23% error correction
                } else if (fmt.id === 'pdf417') {
                    baseOptions.height = 3;     // Row height in modules
                    baseOptions.columns = 10;   // More columns = more capacity
```

Ορίζονται ύψος, πλάτος, έμφαση στο padding για το quiet zone και στο DPI σύμφωνα με τον μαθηματικό τρόπο που αναλύθηκε προηγουμένως για την αποτελεσματικότερη εκτύπωση.

```
    const imgDataUrl = canvas.toDataURL('image/png');
    const width = canvas.width;
    const height = canvas.height;

    // Calculate DPI
    // If scale is 4, 1 module = 4 pixels.
    // For a 10 mil (0.01 inch) module: DPI = 4 / 0.01 = 400.
    // For a 15 mil (0.015 inch) module: DPI = 4 / 0.015 = 267.
    const dpi10mil = Math.round(4 / 0.01);
    const dpi15mil = Math.round(4 / 0.015);
```

Ακολούθως, κρατούνται τα δεδομένα σε μία τιμή newResults τα οποία με τη σειρά τους περνάνε στη τιμή Results μέσω του statement setResults.

```
    newResults.push({
        ...fmt,
        success: true,
        img: imgDataUrl,
        width,
        height,
        dpi10mil,
        dpi15mil,
        readable,
        dataSize
    });
```

Τέλος, τα αποθηκευμένα δεδομένα γίνονται render με τη χρήση mapping για το κάθε format (Aztec, PDF417, QR) όπως φαίνεται παρακάτω.

```
<div className="results-grid">
    {results.map((res) => (
        <div key={res.id} className={`result-card ${res.success ? '' : 'error'}`}>
            <h3>{res.name}</h3>
            {res.success ? (
                <>
                    <div className="barcode-preview">
                        <img src={res.img} alt={res.name} />
                    </div>
                    <div className="stats">
                        <p><strong>Dims:</strong> {res.width} x {res.height} px</p>
                        <p><strong>Rec. DPI (10mil):</strong> {res.dpi10mil}</p>
                        <p><strong>ZXing Readable:</strong> <span className={res.readable ? 'ok' : 'fail'}>{res
                        <p className="meta">Data: {res.dataSize} bytes</p>
                    </div>
                </>
            ) : (
                <div className="error-msg">
                    Failed to generate: {res.error}
                    {res.error.includes("too much data") && " (Payload may exceed capacity)"}
                </div>
            )}
        </div>
    ))}
</div>
```

11. Συμπεράσματα και Προτάσεις

11.1 Επιλογή Τύπου Barcode

Βάσει της ολοκληρωμένης μελέτης, η επιλογή του κατάλληλου τύπου 2D barcode εξαρτάται άμεσα από τη συγκεκριμένη εφαρμογή:

- **QR Code:** Η βέλτιστη γενική επιλογή. Μεγαλύτερη χωρητικότητα (2.953 bytes), ευρύτατη υποστήριξη από smartphones, υποστήριξη Kanji, public domain. Ιδανικό για mobile apps, marketing, authentication.

- **PDF417:** Βέλτιστο για ταξιδιωτικά έγγραφα και ταυτότητες. Η εξαιρετική ανθεκτικότητα (50% recovery) και η δυνατότητα γραμμικής ανάγνωσης το καθιστούν κατάλληλο για τέτοιου είδους εφαρμογές.
- **Aztec:** Κατάλληλο για εισιτήρια μεταφορών. Χωρίς quiet zone = εξοικονόμηση χώρου. Αντέχει σε κακή ποιότητα εκτύπωσης.
- **DataMatrix:** Βέλτιστο για βιομηχανική σήμανση μικρών εξαρτημάτων λόγω υψηλής πυκνότητας.

11.2 Επιλογή Βιβλιοθήκης

Για δημιουργία QR codes: Python: qrcode (απλούστερη, built-in χρώματα). Java: ZXing (πληρέστερη). Kotlin: ZXing. Web: @zxing/library ή qrcode npm.

Για ανάγνωση QR codes: Python: OpenCV ή ZBar. Java: ZXing ή BoofCV. Android: Code Scanner. Web: @zxing/library.

Για multi-format (QR+PDF417+Aztec): ZXing — η μοναδική ολοκληρωμένη λύση. Στο web: bwip-js για δημιουργία, @zxing/library για ανάγνωση.

11.3 Βέλτιστες Πρακτικές Σχεδιασμού

Για τις βέλτιστες πρακτικές σχεδιασμού, προτείνεται M error correction level ως default έτσι ώστε να επιτευχθεί η κατάλληλη ισορροπία στη σχέση μεγέθους-ανθεκτικότητας. Ακόμα, θα ήταν αποτελεσματικότερο να αποφευχθεί η αντιστροφή χρωμάτων και να τηρηθεί το σκούρο foreground με ανοιχτό background. Όσον αφορά το μέγεθος, απαιτείται ως ελάχιστο 2 cm × 2cm και ελάχιστο DPI 300 για εκτύπωση και αναλογία απόστασης μεγέθους δέκα προς ένα (10:1). Για το quiet zone προτείνεται να είναι τέσσερις φορές μεγαλύτερο από το module τουλάχιστον. Τέλος, το svg format είναι το ιδανικό για να επιτευχθεί σημαντικό scalability στα QR Codes και το κάθε QR Code να συνοδεύεται από πληροφορία η οποία σχετίζεται με το περιεχόμενο των κωδικοποιημένων πληροφοριών.

11.4 Περιορισμοί Κωδικοποίησης Εικόνων

Η κωδικοποίηση binary δεδομένων (εικόνων) σε barcodes αντιμετωπίζει ουσιαστικούς περιορισμούς:

- Base64 encoding αυξάνει μέγεθος ~33%
- PDF417 χωρητικότητα ~1.1KB bytes — ανεπαρκής για εικόνες >800 bytes
- QR Code μέγιστο ~2.9KB — αρκετό μόνο για πολύ μικρές εικόνες
- Aztec μέγιστο ~1.9KB — μέτρια χωρητικότητα
- Τα παραγόμενα barcodes είναι πολύ μεγάλα σε pixels

11.5 Συμπέρασμα

Η παρούσα πτυχιακή εργασία αξιολόγησε τα τέσσερα κύρια πρότυπα 2D barcodes (QR Code, PDF417, Aztec, DataMatrix) καθώς και τις σημαντικότερες βιβλιοθήκες λογισμικού ανοιχτού κώδικα για τη δημιουργία και ανάγνωσή τους σε Python, Java και Kotlin. Η έρευνα κατέδειξε ότι κάθε τύπος barcode εξυπηρετεί διαφορετικές ανάγκες: το QR Code αποτελεί τη βέλτιστη γενική επιλογή λόγω μέγιστης χωρητικότητας (2.953 bytes) και υποστήριξης smartphone, το PDF417 υπερέχει σε ταξιδιωτικά έγγραφα με ανθεκτικότητα έως 50%, το Aztec εξοικονομεί χώρο χωρίς quiet zone σε εισιτήρια μεταφορών, και το DataMatrix επικρατεί στη βιομηχανική σήμανση μικρών εξαρτημάτων. Από πλευράς βιβλιοθηκών, η ZXing αναδεικνύεται ως η πληρέστερη λύση καθώς είναι η μοναδική που υποστηρίζει τόσο δημιουργία όσο και ανάγνωση σε 1D και 2D barcodes, ενώ η qrcode (Python), η OpenCV και η BoofCV προσφέρουν native δημιουργία QR codes μέσω των αντίστοιχων APIs τους. Αντιθέτως, οι ZBar και Code Scanner λειτουργούν αποκλειστικά ως αναγνώστες. Τα πειραματικά δεδομένα επιβεβαίωσαν τη σημασία βέλτιστων πρακτικών σχεδιασμού — ελάχιστο μέγεθος 2 cm × 2 cm, αναλογία 10:1 (απόσταση:μέγεθος), quiet zone 4× module, ελάχιστο 300 DPI — καθώς και τους ουσιαστικούς περιορισμούς κωδικοποίησης binary δεδομένων (εικόνων) σε barcodes. Η εργασία αποτελεί οδηγό για χρήστες που επιθυμούν να επιλέξουν τον κατάλληλο τύπο barcode και τη βέλτιστη βιβλιοθήκη ανάλογα με τις απαιτήσεις της εφαρμογής τους.

12. Βιβλιογραφία

12.1 Ιστορία και Εξέλιξη Barcodes

1. Denso Wave. (n.d.). "History of QR Code." QRcode.com — Official QR Code website by Denso Wave. https://www.qrcode.com/en/history/
2. Denso Wave. (n.d.). "What is a QR Code?" QRcode.com. https://www.qrcode.com/en/about/
3. Wikipedia. (2024). "QR Code — History, Design, and Standards." https://en.wikipedia.org/wiki/QR_code
4. ISO/IEC 18004:2015 — Information technology — Automatic identification and data capture techniques — QR Code bar code symbology specification.
5. ISO/IEC 15438:2015 — Information technology — Automatic identification and data capture techniques — PDF417 bar code symbology specification.
6. ISO/IEC 16022:2006 — Information technology — Automatic identification and data capture techniques — Data Matrix bar code symbology specification.
7. ISO/IEC 24778:2008 — Information technology — Automatic identification and data capture techniques — Aztec Code bar code symbology specification.
8. ICAO Doc 9303, Part 3 — Machine Readable Travel Documents — Specifications for Machine Readable Passports (MRPs). https://www.icao.int/publications/pages/publication.aspx?docnum=9303

12.2 Reed-Solomon Error Correction

9. Reed, I. S., & Solomon, G. (1960). "Polynomial Codes Over Certain Finite Fields." Journal of the Society for Industrial and Applied Mathematics, 8(2), 300–304.

12.3 Σύγκριση Τύπων 2D Barcodes

10. Dynamsoft. (2024). "What is the Difference Between QR Code, PDF417 and DataMatrix?" https://www.dynamsoft.com/blog/insights/qr-vs-pdf-417-vs-datamatrix/

11. Scandit. (2024). "Types of Barcodes: Choosing the Right Barcode." https://www.scandit.com/resources/guides/types-of-barcodes-choosing-the-right-barcode/
12. Docutain SDK. (2024). "Barcode Types at a Glance — Overview of Common 1D & 2D Barcode Formats." https://sdk.docutain.com/blogartikel/barcode-types
13. Barcode.design. (2024). "2D Barcode Types Comparison — QR Code vs DataMatrix vs PDF417 vs Aztec." https://barcode.design

12.4 QR Code Usability, Μέγεθος και Εκτύπωση

14. Nielsen Norman Group (NN/g). (2024). "13 QR-Code Usability Guidelines." https://www.nngroup.com/articles/qr-code-guidelines/
15. Triton Store. (2023). "QR Code Size: The Ultimate QR Sizing Guide." https://tritonstore.com.au/qr-code-size/
16. Uniqode. (2024). "QR Code Size: Learn How to Perfectly Size Your QR Codes." https://www.uniqode.com/blog/qr-code-best-practices/how-to-perfectly-size-your-qr-codes
17. IJARSCT. (2024). "Paper on QR Codes — Analysis and Applications." https://ijarsct.co.in/Paper18935.pdf
18. Focardi, R., & Luccio, F. L. (2019). "Usable Security for QR Code." Journal of Information Security and Applications, Elsevier.

12.5 Εφαρμογές σε Βιομηχανία και Μεταφορές

19. Cognex. (2024). "What is an Aztec Code?" https://www.cognex.com/en/tools-and-resources/resource-center/aztec-codes-what-they-are-and-how-they-work

12.6 Βιβλιοθήκες Λογισμικού — Επίσημη Τεκμηρίωση

20. ZXing ("Zebra Crossing") — Open-source multi-format barcode library. https://github.com/zxing/zxing
21. BoofCV — Java Computer Vision Library — QR Code Tutorial. https://boofcv.org/index.php
22. OpenCV. (2024). "QRCodeDetector Class Reference." OpenCV Documentation. https://docs.opencv.org/4.x/de/dc3/classcv_1_1QRCodeDetector.html

23. lincolnloop/python-qrcode — Python QR Code generation library. https://github.com/lincolnloop/python-qrcode
24. pyzbar — Read one-dimensional barcodes and QR codes using the zbar library. https://github.com/NaturalHistoryMuseum/pyzbar
25. yuriy-budiyev/code-scanner — Code Scanner for Android. https://github.com/yuriy-budiyev/code-scanner
26. bwip-js — Barcode Writer in Pure JavaScript. https://github.com/bwipp/

12.7 COVID-19 και Σύγχρονη Υιοθέτηση

27. Juniper Research. (2024). "QR Code Payments: Market Overview and Future Projections." https://www.juniperresearch.com
28. Statista. (2024). "QR Code Usage Worldwide — Statistics & Facts." https://www.statista.com

12.8 Ασφάλεια QR Codes

29. Krombholz, K., Frühwirt, P., Kieseberg, P., Kapsalis, I., Huber, M., & Weippl, E. (2014). "QR Code Security: A Survey of Attacks and Challenges for Usable Security." SBA Research / Vienna University of Technology — Lecture Notes in Computer Science (LNCS), Springer.