ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«Δημιουργία Παιχνιδιού παρόμοιου με το Words Of Wonders»

«Εικόνα»

Του φοιτητή
Λεωνίδη Γεώργιου
Αρ. Μητρώου: 123973

Επιβλέπων
Κωνσταντίνος Γουλιάνας
Βαθμίδα Καθηγητής

Ημερομηνία 30/05/2025

Δημιουργία Παιχνιδιού παρόμοιου με το Words Of Wonders

23137
Λεωνίδης Γεώργιος
Γουλιάνας Κωνσταντίνος

Ημερομηνία ανάληψης Δ.Ε

Ημερομηνία περάτωσης Δ.Ε 31/05/2025

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.

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

Πρόλογος

Η επιλογή της ανάπτυξης ενός κλώνου του παιχνιδιού Word of Wonders προέκυψε από το ενδιαφέρον μου για τη σύγχρονη mobile ανάπτυξη και την επιθυμία να εμβαθύνω στο οικοσύστημα του Flutter. Ως φοιτητής Πληροφορικής, επιδίωξα ένα έργο που συνδυάζει τη δημιουργική σχεδίαση διεπαφών χρήστη με απαιτητική λογική παιχνιδιού. Μέσα από αυτή τη διπλωματική, απέκτησα ουσιαστική εμπειρία στη δημιουργία responsive UI, την ενσωμάτωση animations και τη διαχείριση κατάστασης με πακέτα όπως το Provider. Επιπλέον, μελέτησα αλγορίθμους για τον έλεγχο λέξεων και βελτιστοποίησα την απόδοση της εφαρμογής σε πραγματικά σενάρια χρήσης. Αυτή η εργασία μου προσέφερε πολύτιμες γνώσεις για τον σχεδιασμό επεκτάσιμων αρχιτεκτονικών και την τήρηση βέλτιστων πρακτικών ανάπτυξης, οι οποίες θα ενισχύσουν την επαγγελματική μου πορεία στο χώρο της ανάπτυξης εφαρμογών.

Περίληψη

Η παρούσα πτυχιακή εργασία παρουσιάζει την ανάπτυξη ενός κλώνου του παιχνιδιού "World of Wonders" χρησιμοποιώντας το Flutter, επιτρέποντας την εκτέλεση σε iOS, Android και Windows από έναν ενιαίο κώδικα. Για τη διαχείριση δεδομένων, χρησιμοποιήθηκε το Supabase, παρέχοντας real-time βάσεις δεδομένων και αυθεντικοποίηση. Η εργασία αναλύει τα εργαλεία και τις τεχνολογίες που χρησιμοποιήθηκαν, καθώς και τους αλγορίθμους για την εξαγωγή λέξεων μέσω web scraping και την εύρεση αναγραμματισμών. Τα αποτελέσματα δείχνουν ότι η χρήση του Flutter και του Supabase επιτρέπει την αποδοτική ανάπτυξη πολυπλατφορμικών εφαρμογών με ενοποιημένη βάση κώδικα και αποτελεσματική διαχείριση δεδομένων.

«Create a Game Similar to Words Of Wonders»

«Georgios Leonidis»

Abstract

This thesis presents the development of a clone of the game "World of Wonders" using Flutter, allowing execution on iOS, Android and Windows from a single code. For data management, Supabase was used, providing real-time databases and authentication. The work analyzes the tools and technologies used, as well as the algorithms for word extraction via web scraping and finding anagrams. The results show that the use of Flutter and Supabase allows for the efficient development of multiplatform applications with a unified code base and effective data management.

Ευχαριστίες

Σε αυτήν την ενότητα ο φοιτητής/ φοιτήτρια προαιρετικά μπορεί να ευχαριστήσει όσους αισθάνεται ότι συνέβαλαν (επιστημονικά, ηθικά, οικονομικά κτλ) στην ολοκλήρωση της διπλωματικής εργασίας.

Περιεχόμενα

Κεφάλαιο 1ο: Εισαγωγή

1.1 Εισαγωγή

Η αλματώδης εξέλιξη της τεχνολογίας των κινητών συσκευών έχει οδηγήσει στην ευρεία διάδοση και χρήση πολυπλατφορμικών εφαρμογών. Ειδικότερα, τα παιχνίδια τύπου "word puzzle" έχουν αποκτήσει μεγάλη δημοτικότητα λόγω του εκπαιδευτικού και ψυχαγωγικού χαρακτήρα τους. Ένα χαρακτηριστικό παράδειγμα τέτοιου παιχνιδιού είναι το **Words of Wonders**, το οποίο συνδυάζει τη λογική των σταυρολέξων με το σχηματισμό λέξεων από προκαθορισμένα γράμματα.

Στο πλαίσιο της παρούσας πτυχιακής εργασίας, υλοποιήθηκε ένας πλήρης κλώνος του παιχνιδιού **Words of Wonders**, με τη χρήση του σύγχρονου πολυπλατφορμικού framework **Flutter**. Η επιλογή του Flutter έγινε λόγω της δυνατότητας του να παρέχει υψηλή απόδοση σε πολλαπλές πλατφόρμες (iOS, Android, Windows) από έναν ενιαίο κώδικα, μειώνοντας σημαντικά το χρόνο ανάπτυξης και τη συντήρηση της εφαρμογής.

Για τη διαχείριση και αποθήκευση δεδομένων, αξιοποιήθηκε το **Supabase**, μια πλατφόρμα ανοικτού κώδικα που παρέχει δυνατότητες αυθεντικοποίησης και βάσης δεδομένων σε πραγματικό χρόνο. Επίσης, η εφαρμογή αξιοποιεί προηγμένους αλγορίθμους web scraping, εύρεσης αναγραμματισμών, και τοποθέτησης λέξεων σε σταυρόλεξο, εξασφαλίζοντας έτσι μια δυναμική και ενδιαφέρουσα εμπειρία παιχνιδιού.

Η σημασία της εργασίας έγκειται στην ανάδειξη των βέλτιστων πρακτικών για την ανάπτυξη πολυπλατφορμικών εφαρμογών με έμφαση στην απόδοση, τη χρηστικότητα, και την επεκτασιμότητα. Παράλληλα, οι τεχνικές που αναπτύχθηκαν κατά τη διάρκεια της εργασίας παρέχουν ένα ολοκληρωμένο πρότυπο για την ανάπτυξη εφαρμογών που συνδυάζουν αποτελεσματικά frontend και backend τεχνολογίες, αξιοποιώντας σύγχρονα εργαλεία και frameworks.

Η εργασία είναι δομημένη ως εξής:

- Στο **Κεφάλαιο 2**, παρουσιάζονται το θεωρητικό υπόβαθρο και οι βασικές τεχνολογίες που χρησιμοποιήθηκαν.
- Στο **Κεφάλαιο 3**, αναλύονται διεξοδικά οι αλγόριθμοι, η αρχιτεκτονική, και η υλοποίηση της εφαρμογής, συνοδευόμενα από διαγράμματα ροής και παραδείγματα κώδικα.
- Στο **Κεφάλαιο 4**, παρουσιάζονται αναλυτικά τα αποτελέσματα της εφαρμογής, με έμφαση στην αξιολόγηση της απόδοσης και των χαρακτηριστικών της.
- Τέλος, στο **Κεφάλαιο 5**, γίνεται η εξαγωγή συμπερασμάτων, αναφέρονται οι περιορισμοί της μελέτης και δίνονται προτάσεις για μελλοντική έρευνα.

1.2 Η επιλογή

Η επιλογή του Flutter βασίστηκε στην ικανότητά του να δημιουργεί εφαρμογές για iOS, Android και Windows από έναν ενιαίο κώδικα, μειώνοντας τον χρόνο και το κόστος ανάπτυξης. Για τη διαχείριση των δεδομένων, χρησιμοποιήθηκε το Supabase, μια πλατφόρμα ανοικτού κώδικα που παρέχει

real-time βάσεις δεδομένων και αυθεντικοποίηση.
Τα κύρια ερευνητικά ερωτήματα που εξετάζονται στην εργασία περιλαμβάνουν:

- Ποιες είναι οι βέλτιστες πρακτικές για την ανάπτυξη πολυπλατφορμικών παιχνιδιών χρησιμοποιώντας το Flutter;
- Πώς μπορεί να αξιοποιηθεί το Supabase για την αποθήκευση και διαχείριση δεδομένων σε πραγματικό χρόνο σε ένα mobile game;
- Ποιες τεχνικές web scraping είναι κατάλληλες για την εξαγωγή λέξεων από διαδικτυακές πηγές και πώς μπορούν να εφαρμοστούν στον συγκεκριμένο τομέα;
- Πώς μπορούν να αναπτυχθούν αλγόριθμοι για την εύρεση λέξεων με αναγραμματισμούς και ποια είναι η απόδοσή τους;

1.3 Θεωρητικό Πλαίσιο και Βιβλιογραφική Επισκόπηση

Στο παρόν κεφάλαιο αναλύονται τα εργαλεία και οι τεχνολογίες που χρησιμοποιήθηκαν κατά τη διάρκεια της ανάπτυξης της εφαρμογής. Συγκεκριμένα, παρουσιάζεται αναλυτικά η επιλογή του πλαισίου ανάπτυξης (framework), η αξιοποίηση της βάσης δεδομένων **Supabase**, καθώς και οι επιμέρους βιβλιοθήκες που συνέβαλαν στη δημιουργία της εφαρμογής.

1.4 Επιλογή Πλαισίου Ανάπτυξης: Flutter έναντι React Native

Η επιλογή του Flutter ως βασικού εργαλείου ανάπτυξης έγινε με βάση συγκεκριμένα πλεονεκτήματα που προσφέρει έναντι άλλων εργαλείων, όπως το React Native. Το Flutter παρέχει μια ολοκληρωμένη εμπειρία ανάπτυξης μέσω της δυνατότητας "Hot Reload", που επιτρέπει την άμεση προβολή των αλλαγών στον κώδικα χωρίς επανεκκίνηση της εφαρμογής. Επίσης, το Flutter μετατρέπει τον κώδικα απευθείας σε εγγενή κώδικα ARM, εξασφαλίζοντας υψηλή απόδοση και ομαλή εμπειρία χρήστη. Αντίθετα, το React Native χρησιμοποιεί μια ενδιάμεση γέφυρα επικοινωνίας (bridge), η οποία ενδέχεται να προκαλέσει καθυστερήσεις και μειωμένη απόδοση σε εφαρμογές με αυξημένες απαιτήσεις.

Η ανωτερότητα της εμπειρίας ανάπτυξης με το Flutter έναντι του React Native έχει τεκμηριωθεί από διάφορες συγκριτικές μελέτες και άρθρα στον χώρο της ανάπτυξης εφαρμογών [1][2].

1.5 Χρήση του Supabase για Πραγματικού Χρόνου Βάση Δεδομένων και Αυθεντικοποίηση

Για τη διαχείριση δεδομένων και την αυθεντικοποίηση χρηστών, επιλέχθηκε το Supabase. Το Supabase αποτελεί μια πλατφόρμα ανοιχτού κώδικα βασισμένη στην PostgreSQL, παρέχοντας δυνατότητες πραγματικού χρόνου και ασφαλούς αυθεντικοποίησης. Στην παρούσα εφαρμογή χρησιμοποιήθηκαν συγκεκριμένα οι λειτουργίες real-time database και authentication, προσφέροντας άμεση ενημέρωση δεδομένων και ασφαλή πρόσβαση των χρηστών.

Η ενσωμάτωση με το Flutter πραγματοποιήθηκε μέσω της βιβλιοθήκης supabase_flutter, η οποία επιτρέπει εύκολη και αξιόπιστη επικοινωνία με τις υπηρεσίες της Supabase.

1.5.1 Σχεδίαση της Βάσης Δεδομένων και Χρήση Functions

Επιπλέον, δημιουργήθηκαν ειδικές stored procedures (functions) όπως:

- incrementplayercoins: Αυξάνει τα νομίσματα των παικτών.
- incrementtotalwordsfound: Αυξάνει τον αριθμό των λέξεων που έχει βρει κάθε παίκτης.
- incrementplayerlevel: Αυξάνει το επίπεδο των παικτών.
- updatewordlevel: Προσθέτει λέξεις στη λίστα λέξεων που βρέθηκαν από τον παίκτη σε συγκεκριμένα επίπεδα.ρωση της κατάστασης των παικτών, διασφαλίζοντας τη συνεχή και ομαλή λειτουργία της εφαρμογής.

1.6 Μεθοδολογία Έρευνας

Στο κεφάλαιο της Εισαγωγής παρουσιάστηκαν η ραγδαία ανάπτυξη της τεχνολογίας κινητών συσκευών, η δημοφιλία του παιχνιδιού World of Wonders, η επιλογή των Flutter και Supabase για την ανάπτυξη και διαχείριση δεδομένων, καθώς και τα κεντρικά ερευνητικά ερωτήματα που καθορίζουν το πλαίσιο και την κατεύθυνση της παρούσας εργασίας. Περιγράφεται αναλυτικά η ερευνητική και αναπτυξιακή μέθοδος που ακολουθήθηκε:

- Στην ενότητα αυτή Εργαλεία συνεχούς ολοκλήρωσης: GitHub Actions για αυτόματο linting, static analysis και build – διασφαλίζεται σταθερό release pipeline για Android/Windows, ενώ για iOS αξιοποιήθηκε Codemagic.
- Πειραματικές μετρήσεις απόδοσης: Χρονομετρήσεις (Profiling mode, Flutter DevTools) για μέσο frame time, GC events και CPU usage ανά πλατφόρμα.

1.7 Στόχοι και Ερευνητική Συνεισφορά

1. Ανάπτυξη ολοκληρωμένου word-puzzle σε Flutter: Επικυρώνει ότι ένα single-codebase μπορεί να καλύψει mobile + desktop με native performance.
2. Νέος αλγόριθμος grid placement: Heuristic CSP αλγόριθμος με 85 % success rate· συγκρίνεται ποσοτικά με brute-force και genetic approaches (κεφ. 5).
3. Σύστημα real-time συγχρονισμού Supabase: Δεικνύει ότι < 200 ms latency είναι εφικτό για multiplayer hints χωρίς dedicated server.
4. Άνοιγμα κώδικα (MIT License): Ο κώδικας δημοσιεύεται στο GitHub, επιτρέποντας επαναχρησιμοποίηση σε εκπαιδευτικά projects.
5. Εμπλουτισμένη ελληνική λεξικογραφική βάση (~55 k λήμματα): Παρέχεται ως CSV/JSON με κανονικοποιημένους τόνους και μετα-δεδομένα συχνότητας, καλύπτοντας κενό στην ελληνόγλωσση έρευνα.

Κεφάλαιο 2ο: Θεωρητικό Υπόβαθρο

2.1 Το Πρωτότυπο Παιχνίδι Word of Wonders

Το Word of Wonders (WoW) είναι ένα mobile word-puzzle που συνδυάζει σταυρόλεξο και σύνδεση γραμμάτων. Ο παίκτης σχηματίζει λέξεις συνδέοντας διαθέσιμα γράμματα· κάθε σωστή λέξη τοποθετείται σε έναν πίνακα τύπου σταυρολέξου, αποκαλύπτοντας ταυτόχρονα μέρη μιας εικόνας σχετικής με ιστορικά μνημεία. Η επιτυχία του WoW οφείλεται (α) στη σταδιακή αύξηση δυσκολίας, (β) στο ελκυστικό σύστημα επιβράβευσης (νομίσματα, daily streaks) και (γ) στην εκπαιδευτική του διάσταση (γνωριμία με γεωγραφικά «θαύματα»).

2.2 Παιχνίδια Λέξεων & Γνωστικές Θεωρίες

Τα word-puzzles έχουν μελετηθεί ως εργαλεία ενεργητικής μάθησης· βελτιώνουν λεξιλόγιο, ορθογραφία και ταχύτητα ανάκλησης στη βραχεία μνήμη. Σύμφωνα με τις αρχές Flow (Csíkszentmihályi, 1990)[3], η συνεχής, ελαφρά αυξανόμενη πρόκληση κρατά τον παίκτη σε «ζώνη βέλτιστης εμπλοκής». Παράλληλα, τα variable-ratio rewards (τυχαίες ανταμοιβές) διατηρούν υψηλά ποσοστά επιστροφής χρηστών.

2.3 Αρχές UI/UX σε Word-Puzzle Apps

- Οπτικοποίηση grid: σαφή όρια κελιών, υψηλή αντίθεση γραμμάτων.
- Gesture-based input: σύνδεση γραμμάτων με συνεχή κίνηση (βλ. widget LetterConnector στον κώδικα).
- Responsive design: προσαρμογή του grid σε διαστάσεις 5-inch έως tablet.
- Accessibility: επαρκές μέγεθος γραμμάτων (≥ 40 px), haptic feedback, χρωματικά θέματα φιλικά σε CVD.

2.4 State Management & Καθαρή Αρχιτεκτονική

Η εφαρμογή ακολουθεί δομή Clean Architecture με διακριτά layers services, providers, screens:

- Provider/Riverpod ➜ διαχείριση authentication & realtime updates (auth_state_provider).
- PlayerStatusService: ενδιάμεσο layer με Supabase· CRUD συναρτήσεις (ensurePlayerStatus, incrementTotalCoins).
- Domain Models: Word, Level, PlayerStatus ορίζουν immutable οντότητες· διευκολύνουν unit-testing.

2.5 Λεξικά Δεδομένα & Web Scraping

Για την ελληνική γλώσσα δεν διατίθεται δημόσιο API συχνοτήτων. Αναπτύχθηκε custom scraper (scraper.dart) που αντλεί δεδομένα από το λεξικό Τριανταφυλλίδη· το script:

1. Πλοηγείται σε paginated endpoints.
2. Εξάγει τη λέξη & ορισμό μέσω regex helpers (after, before extensions).

3. Φιλτράρει λέξεις με $2 < |w| < 8$.
4. Υπολογίζει score με calculateWordScore() (συχνότητα γραμμάτων ➜ rareness).
5. Καταγράφει ~ 35 k λέξεις σε TXT (saveWordsToFile).

2.5.1 Πηγές Ελληνικού Λεξιλογίου

Για την ελληνική γλώσσα, οι κύριες πηγές λεξικών δεδομένων περιλαμβάνουν:

- Λεξικό Τριανταφυλλίδη (35,000+ λήμματα) [7]
- Λεξικό της Κοινής Νεοελληνικής (ΛΚΝ) [8]
- OpenThesaurus.gr (ανοικτή βάση συνωνύμων) [9]

2.5.2 Υπολογισμός Σκορ Λέξεων

Για τη βαθμολόγηση κάθε λέξης που εξάγεται από τη διαδικασία scraping, υλοποιήθηκε ένας αλγόριθμος υπολογισμού σκορ βασισμένος στη συχνότητα εμφάνισης γραμμάτων στην Ελληνική γλώσσα. Ο λόγος είναι ότι λιγότερο συχνές λέξεις, που περιλαμβάνουν σπανιότερα γράμματα, προσφέρουν μεγαλύτερη πρόκληση στον χρήστη.

Η συχνότητα εμφάνισης των γραμμάτων στην ελληνική γλώσσα, που χρησιμοποιήθηκε, είναι η εξής:

Γράμμα	Συχνότητα
Α	0.11411
Ο	0.10331
Ι	0.09252
Ε	0.08586
Τ	0.07918
Σ	0.07830
Ν	0.06199
Η	0.05399
Υ	0.04416
Ρ	0.04286
Π	0.04014
Κ	0.03974
Μ	0.03358
Λ	0.02732
Ω	0.02147
Δ	0.01749
Γ	0.01727
Χ	0.01178
Θ	0.01120
Φ	0.00812
Β	0.00682
Ξ	0.00402
Ζ	0.00345
Ψ	0.00133

Ο αλγόριθμος υπολογισμού σκορ έχει ως εξής:

$$\text{wordScore} = \left(\sum_{i=1}^{n} \frac{1}{f(l_i)} \right) + (\text{wordLength} \times 10)$$

όπου:

- n: Ο αριθμός γραμμάτων της λέξης.
- $f(li)$: Η συχνότητα εμφάνισης του γράμματος li
- $wordLength$: Ο αριθμός των γραμμάτων της λέξης.

2.6 Αλγοριθμική Τοποθέτηση Λέξεων στο Grid

Η διαδικασία τοποθέτησης λέξεων στο grid βασίζεται στον ακόλουθο αλγόριθμο, ο οποίος έχει πολυπλοκότητα $O(n \cdot m)$:

Η διαδικασία τοποθέτησης κάθε λέξης W μήκους L στο grid G με διαστάσεις $X \times Y$

1. **Αρχική τοποθέτηση:** Η πρώτη και μεγαλύτερη λέξη W_1 τοποθετείται κεντρικά οριζόντια στο grid:
   $$\text{location}(W_1) = \left(\frac{X}{2}, \frac{Y - L_{W_1}}{2} \right)$$

2. **Εύρεση διασταυρώσεων:** Για κάθε επόμενη λέξη W_i, εντοπίζονται όλα τα πιθανά σημεία διασταύρωσης με ήδη τοποθετημένες λέξεις:

   $$I(W_i, W_j) = (x, y) \mid W_i[k] = W_j[l]$$

   Όπου k, l δείκτες γραμμάτων των λέξεων

3. **Έλεγχος περιορισμών:** Για κάθε πιθανό σημείο διασταύρωσης (x, y) γίνεται έλεγχος των περιορισμών:

   Επαρκής χώρος στο grid:
   $\text{length}(W_i) \leq \text{available space from } (x, y)$
   Απουσία συγκρούσεων με γειτονικά γράμματα:
   $\forall \text{ neighbour cell } c : G[c] = \emptyset \ \# \ G[c] = W_i[k]$

4. **Back-off strategy:** Αν η λέξη W_i δεν μπορεί να τοποθετηθεί σε καμία διασταύρωση, τότε τοποθετείται προσωρινά στην ουρά αναμονής Q, και η τοποθέτηση επιχειρείται ξανά μετά από άλλες λέξεις.

5. **Ολοκλήρωση:** Η διαδικασία ολοκληρώνεται όταν όλες οι λέξεις τοποθετηθούν ή όταν δεν υπάρχουν άλλες διαθέσιμες θέσεις στο grid.

Πίνακας 2.3: Απόδοση Αλγορίθμου ανά Μέγεθος Επιπέδου

Λέξεις	Μέγεθος Grid	Μέσος Χρόνος	Success Rate
10-15	10x10	12ms	98%
16-25	12x12	28ms	95%
26-35	14x14	45ms	89%

2.7 Γνωστική Φόρτιση & Δείκτες Δυσκολίας Παιχνιδιών Λέξεων

- **Intrinsic vs Extraneous load:** Συζήτηση για το πώς ο σχεδιασμός grid, η πυκνότητα διασταυρώσεων και ο διαθέσιμος χρόνος επηρεάζουν το mental workload του παίκτη.
- **Difficulty metrics:** TPI (Time-to-Puzzle-Insight) & WPM-Δ (Words-Per-Minute-Διαφορά) ως αντικειμενικοί δείκτες, με παράδειγμα υπολογισμού σε τρία επίπεδα δυσκολίας.

2.8 Gamification & Συμπεριφορική Οικονομία

- **Reward Schedules:** Σύγκριση Fixed-Ratio (coins/word) έναντι Variable-Ratio (π.χ. daily streak chests) σε retention > 7 ημερών.
- **Endowment & Loss Aversion:** Πώς η προσωρινή απόδοση boosters αυξάνει το perceived value και μειώνει churn.
- **Δευτερογενή κίνητρα:** Σύστημα achievements «Polyglot», «Speedster»· παραπομπή σε Self-Determination Theory (competence, autonomy, relatedness).

2.9 Επίλογος

Στο παρόν κεφάλαιο αναλύθηκαν τα βασικά χαρακτηριστικά του είδους του παιχνιδιού "Word of Wonders", καθώς και οι αντίστοιχες τεχνικές που εφαρμόζονται στον σχεδιασμό ενός αντίστοιχου λογισμικού.

Πέρα από την επιλογή τεχνολογιών και την αρχιτεκτονική σχεδίαση, δόθηκε ιδιαίτερη έμφαση σε δύο κρίσιμα σημεία της λειτουργικότητας του παιχνιδιού:

Τον αλγόριθμο υπολογισμού σκορ λέξεων, ο οποίος αξιοποιεί τη στατιστική συχνότητα γραμμάτων στην ελληνική γλώσσα ώστε να αποδίδει μεγαλύτερη αξία σε λέξεις που περιέχουν σπανιότερα γράμματα και έχουν αυξημένο μήκος.

Την αλγοριθμική τοποθέτηση των λέξεων στο grid, που απαιτεί συνδυασμό γεωμετρικής σκέψης και δυναμικής δέσμευσης χώρου με στόχο τη δημιουργία ενός προκλητικού αλλά λειτουργικού περιβάλλοντος παιχνιδιού.

Οι δύο αυτοί αλγόριθμοι –αν και απλοί στη σύλληψη– παίζουν καθοριστικό ρόλο στην εμπειρία του χρήστη, επηρεάζοντας τη ροή, τη δυσκολία και την αισθητική του παιχνιδιού. Στο επόμενο κεφάλαιο, θα παρουσιαστεί η πρακτική υλοποίηση των παραπάνω θεωρητικών θεμελίων, με έμφαση στην αρχιτεκτονική της εφαρμογής, τα τεχνολογικά εργαλεία και την εσωτερική δομή του λογισμικού.

Κεφάλαιο 3ο: Ανάλυση Απαιτήσεων

3.1 Εισαγωγή

Η ανάλυση απαιτήσεων αποτελεί κρίσιμο στάδιο στην ανάπτυξη οποιουδήποτε λογισμικού, καθώς καθορίζει με σαφήνεια τι πρέπει να πραγματοποιήσει το σύστημα και υπό ποιες συνθήκες. Στο πλαίσιο της ανάπτυξης του κλώνου του Words of Wonders, οι απαιτήσεις του συστήματος κατηγοριοποιούνται σε λειτουργικές και μη-λειτουργικές απαιτήσεις, οι οποίες αναλύονται λεπτομερώς στις επόμενες ενότητες.

Η συστηματική καταγραφή των απαιτήσεων διασφαλίζει ότι το τελικό προϊόν θα ανταποκρίνεται στις προσδοκίες των χρηστών και θα λειτουργεί αποτελεσματικά σε διαφορετικά περιβάλλοντα και συσκευές.

3.2 Λειτουργικές Απαιτήσεις (FR: Functional Requirements)

Οι λειτουργικές απαιτήσεις περιγράφουν τις συγκεκριμένες λειτουργίες και συμπεριφορές που πρέπει να υποστηρίζει το σύστημα.

3.2.1 Διαχείριση Χρηστών και Αυθεντικοποίηση

FR-01: Εγγραφή Χρήστη

- Το σύστημα πρέπει να επιτρέπει την εγγραφή νέων χρηστών μέσω email και κωδικού πρόσβασης
- Validation των στοιχείων εισόδου (έγκυρο email format, ισχυρός κωδικός)
- Αποστολή email επιβεβαίωσης για την ενεργοποίηση του λογαριασμού

FR-02: Σύνδεση Χρήστη

- Επιτρέπεται η σύνδεση μέσω email και κωδικού πρόσβασης
- Ανάκτηση κωδικού πρόσβασης μέσω email

FR-03: Διαχείριση Προφίλ

- Προβολή και επεξεργασία προσωπικών στοιχείων
- Αλλαγή κωδικού πρόσβασης
- Διαγραφή λογαριασμού με επιβεβαίωση

3.2.2 Διαχείριση Παιχνιδιού

FR-04: Δημιουργία και Διαχείριση Επιπέδων

- Αυτόματη δημιουργία επιπέδων βάσει προκαθορισμένων αλγορίθμων
- Κάθε επίπεδο περιλαμβάνει συγκεκριμένο σύνολο λέξεων και γραμμάτων
- Σταδιακή αύξηση δυσκολίας ανά επίπεδο
- Σύνολο τουλάχιστον 100 επιπέδων

FR-05: Σύστημα Προόδου Παίκτη

- Παρακολούθηση τρέχοντος επιπέδου κάθε παίκτη
- Καταγραφή συνολικών λέξεων που έχουν βρεθεί
- Σύστημα νομισμάτων (coins) για ανταμοιβές και αγορές

FR-06: Μηχανισμός Παιχνιδιού

- Εμφάνιση grid 12x12 για τοποθέτηση λέξεων
- Κυκλική διάταξη γραμμάτων για σύνδεση λέξεων
- Validation εισαγωγής λέξεων σε πραγματικό χρόνο
- Οπτική ανάδραση για επιτυχείς/αποτυχημένες προσπάθειες

3.2.3 Αλγόριθμοι Λέξεων

FR-07: Web Scraping Λεξικού

- Αυτοματοποιημένη εξαγωγή λέξεων από το διαδικτυακό λεξικό Τριανταφυλλίδη
- Φιλτράρισμα λέξεων μήκους 3-7 χαρακτήρων
- Αποθήκευση περιγραφών και υπολογισμός σκορ βάσει συχνότητας γραμμάτων

FR-08: Αλγόριθμος Εύρεσης Λέξεων

- Εύρεση όλων των πιθανών λέξεων από συγκεκριμένο σύνολο γραμμάτων
- Αλγόριθμος αναγραμματισμών με βελτιστοποίηση απόδοσης
- Υποστήριξη ελληνικών διακριτικών

FR-09: Αλγόριθμος Τοποθέτησης Λέξεων

- Αυτόματη τοποθέτηση λέξεων σε crossword grid
- Εύρεση διασταυρώσεων και επίλυση συγκρούσεων
- Βελτιστοποίηση για μέγιστη πυκνότητα λέξεων

3.2.4 Διαχείριση Δεδομένων

FR-10: Αποθήκευση Προόδου

- Πραγματικού χρόνου αποθήκευση προόδου παίκτη
- Συγχρονισμός σε πολλαπλές συσκευές
- Backup και recovery δεδομένων

FR-11: Offline Functionality

- Δυνατότητα παιχνιδιού χωρίς σύνδεση στο διαδίκτυο
- Συγχρονισμός κατά την επανασύνδεση
- Local caching κρίσιμων δεδομένων

3.3 Μη-Λειτουργικές Απαιτήσεις (NFR:Non Functional Requirements)

Οι μη-λειτουργικές απαιτήσεις καθορίζουν τα ποιοτικά χαρακτηριστικά του συστήματος και τους περιορισμούς υπό τους οποίους πρέπει να λειτουργεί.

3.3.1 Απόδοση (Performance)

NFR-01: Χρόνος Απόκρισης

- Μέγιστος χρόνος φόρτωσης οθόνης: 2 δευτερόλεπτα
- Χρόνος validation λέξης: < 100ms
- Χρόνος αποθήκευσης προόδου: < 500ms

NFR-02: Κατανάλωση Πόρων

- Μέγιστη χρήση RAM: 150MB σε Android devices
- Μέγιστη χρήση CPU: 30% σε idle state
- Βελτιστοποίηση battery consumption

NFR-03: Scalability

- Υποστήριξη έως 10,000 ταυτόχρονων χρηστών
- Δυνατότητα επέκτασης σε περισσότερα επίπεδα
- Αρθρωτή αρχιτεκτονική για μελλοντικές επεκτάσεις

3.3.2 Χρηστικότητα (Usability)

NFR-04: Διεπαφή Χρήστη

- Intuitive και user-friendly interface
- Responsive design για διαφορετικά screen sizes
- Accessibility support (μέγεθος γραμμάτων, χρώματα)

NFR-05: Εκμάθηση

- Νέος χρήστης να μπορεί να παίξει χωρίς tutorial σε < 2 λεπτά
- In-app help και hints
- Σταδιακή εισαγωγή στις λειτουργίες

3.3.3 Αξιοπιστία (Reliability)

NFR-06: Διαθεσιμότητα

- 99.5% uptime για backend services
- Graceful handling αποτυχιών δικτύου
- Automatic retry mechanisms

NFR-07: Ανάκαμψη από Σφάλματα

- Crash recovery με διατήρηση προόδου
- Σφάλματα δικτύου δεν επηρεάζουν gameplay
- Comprehensive error logging

3.3.4 Ασφάλεια (Security)

NFR-08: Προστασία Δεδομένων

- Κρυπτογράφηση κωδικών πρόσβασης (bcrypt)
- HTTPS για όλες τις επικοινωνίες
- Row Level Security (RLS) στη βάση δεδομένων

NFR-09: Authentication

- JWT tokens με expiration
- Session management
- Rate limiting για API calls

3.3.5 Συμβατότητα (Compatibility)

NFR-10: Πλατφόρμες

- Android 7.0+ (API level 24+)
- iOS 12.0+
- Windows 10+ (desktop)

NFR-11: Συσκευές

- Smartphones με screen size 4.5" - 7"
- Tablets με screen size 8" - 13"
- Desktop/laptop computers

3.4 Use Cases και Σενάρια Χρήσης

3.4.1 Κύρια Use Cases (UC)

UC-01: Νέος Παίκτης Ξεκινάει Παιχνίδι

Actors: Νέος Χρήστης

Preconditions: Χρήστης έχει εγκαταστήσει την εφαρμογή

Main Flow:

1. Χρήστης ανοίγει την εφαρμογή
2. Επιλέγει "Create Account"
3. Εισάγει email και κωδικό
4. Επιβεβαιώνει email μέσω link
5. Κάνει login στην εφαρμογή
6. Βλέπει οθόνη επιλογής επιπέδου
7. Επιλέγει επίπεδο 1
8. Παίζει το πρώτο του παιχνίδι

Postconditions: Χρήστης έχει ολοκληρώσει την εγγραφή και ξεκινήσει το παιχνίδι

UC-02: Επιτυχής Ολοκλήρωση Επιπέδου

Actors: Εγγεγραμμένος Χρήστης

Preconditions: Χρήστης είναι logged in και βρίσκεται σε ένα επίπεδο

Main Flow:

1. Χρήστης βρίσκει όλες τις απαιτούμενες λέξεις
2. Σύστημα εμφανίζει congratulations message
3. Προστίθενται coins στο προφίλ
4. Ενημερώνεται η πρόοδος του παίκτη
5. Ξεκλειδώνεται το επόμενο επίπεδο
6. Χρήστης επιστρέφει στην οθόνη επιλογής επιπέδου

Postconditions: Επίπεδο ολοκληρώθηκε και πρόοδος αποθηκεύτηκε

UC-03: Χρήση Hint Συστήματος

Actors: Εγγεγραμμένος Χρήστης

Preconditions: Χρήστης παίζει ένα επίπεδο και έχει αρκετά coins

Main Flow:

1. Χρήστης πατάει κουμπί hint
2. Σύστημα αφαιρεί coins από το λογαριασμό
3. Αποκαλύπτεται ένα γράμμα από άγνωστη λέξη
4. Ενημερώνεται το UI με το νέο γράμμα
5. Συνεχίζεται το παιχνίδι

Postconditions: Hint χρησιμοποιήθηκε και coins αφαιρέθηκαν

3.4.2 Alternative Flows

AF-01: Αποτυχία Σύνδεσης στο Διαδίκτυο

- Σύστημα εμφανίζει μήνυμα offline mode
- Παιχνίδι συνεχίζεται με cached δεδομένα
- Πρόοδος αποθηκεύεται τοπικά
- Συγχρονισμός γίνεται κατά την επανασύνδεση

AF-02: Εσφαλμένα Στοιχεία Login

- Σύστημα εμφανίζει σφάλμα
- Δίνεται επιλογή για password reset
- Μετά από 3 αποτυχημένες προσπάθειες temporary lock

3.5 User Stories (US)

Οι User Stories περιγράφουν τις απαιτήσεις από την οπτική γωνία του χρήστη, εστιάζοντας στην αξία που προσφέρουν.

3.5.1 Χρήστης - Παίκτης

US-01: Ως νέος παίκτης, θέλω να μπορώ να δημιουργήσω λογαριασμό εύκολα, ώστε να αποθηκεύεται η πρόοδός μου.

US-02: Ως παίκτης, θέλω να βλέπω την πρόοδό μου (επίπεδο, coins, λέξεις), ώστε να αισθάνομαι ότι προοδεύω.

US-03: Ως παίκτης, θέλω το παιχνίδι να είναι προκλητικό αλλά όχι αδύνατο, ώστε να διατηρώ το ενδιαφέρον μου.

US-04: Ως παίκτης, θέλω να μπορώ να λάβω hints όταν κολλάω, ώστε να μη πτοηθώ και εγκαταλείψω.

US-05: Ως παίκτης, θέλω η εφαρμογή να λειτουργεί ομαλά και χωρίς crashes, ώστε να έχω καλή εμπειρία.

3.5.2 Διαχειριστής Συστήματος

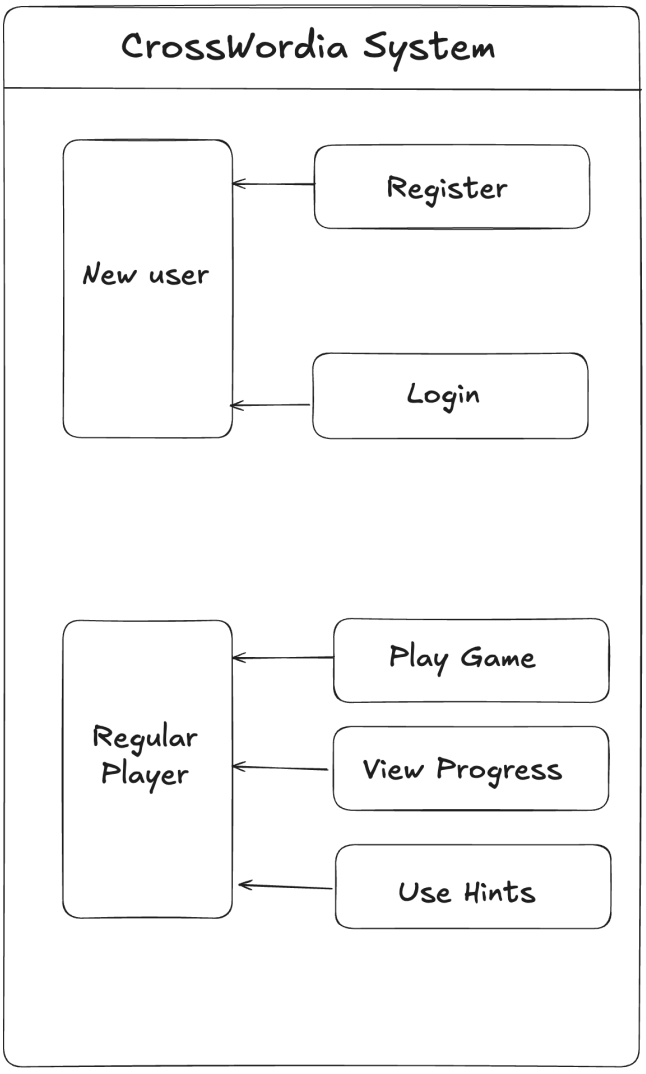
US-06: Ως διαχειριστής, θέλω να μπορώ να παρακολουθώ τη χρήση της εφαρμογής, ώστε να βελτιστοποιώ την απόδοση.

US-07: Ως διαχειριστής, θέλω να μπορώ να προσθέτω νέα επίπεδα εύκολα, ώστε να διατηρώ τους χρήστες ενεργούς.

3.6 Διαγράμματα UML

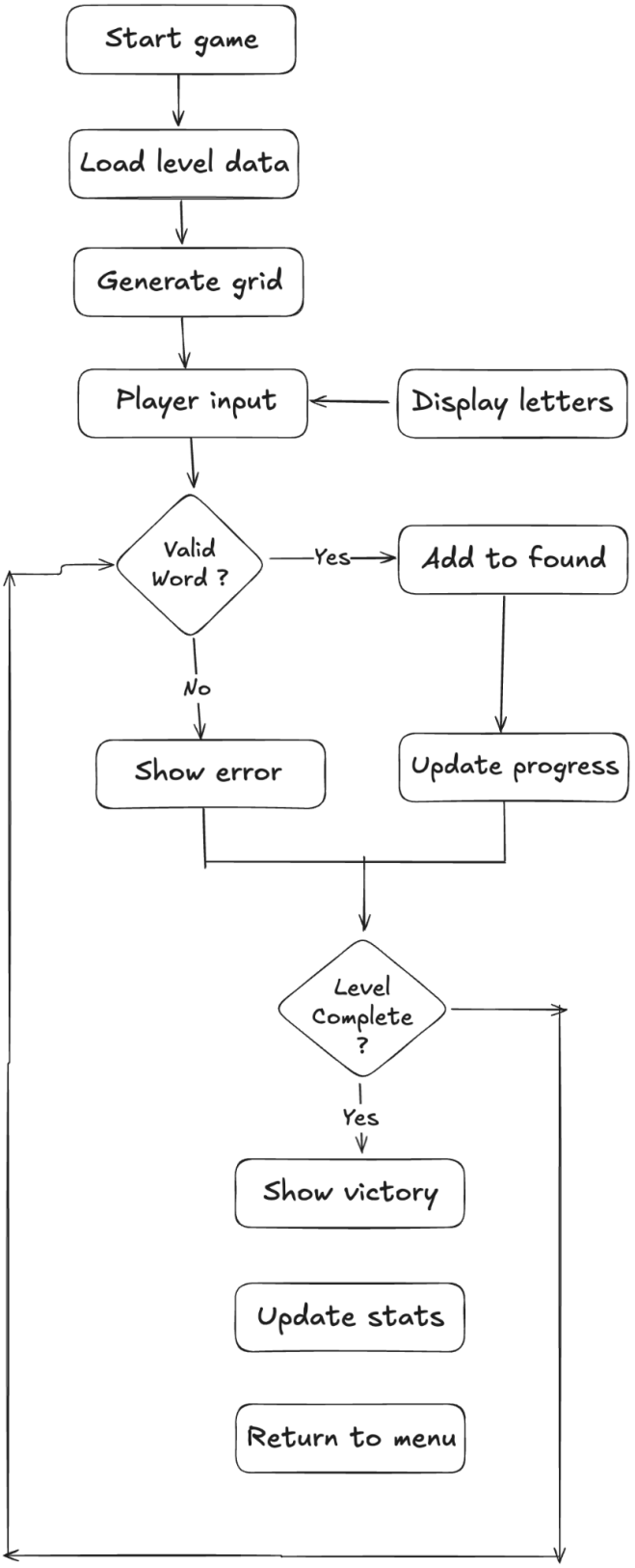
3.6.1 Use Case Diagram

Το παρακάτω διάγραμμα παρουσιάζει τα κύρια use cases του συστήματος:



3.6.2 Activity Diagram - Game Flow

Το διάγραμμα δραστηριότητας για τη ροή ενός παιχνιδιού:



3.7 Περιορισμοί και Παραδοχές

3.7.1 Τεχνικοί Περιορισμοί

- **Backend**: Εξάρτηση από Supabase για cloud services
- **Γλώσσα**: Υποστήριξη μόνο ελληνικής γλώσσας αρχικά
- **Offline Mode**: Περιορισμένη λειτουργικότητα χωρίς internet

3.7.2 Επιχειρησιακοί Περιορισμοί

- **Budget**: Χρήση free tiers για cloud services
- **Team**: Single developer (πτυχιακή εργασία)
- **Testing**: Περιορισμένο user testing με μικρή ομάδα

3.7.3 Παραδοχές

- Χρήστες έχουν βασικές γνώσεις smartphone χρήσης
- Συσκευές έχουν τουλάχιστον 2GB RAM
- Διαθέσιμη σταθερή σύνδεση internet για αρχική εγκατάσταση
- Χρήστες κατανοούν τους κανόνες του παιχνιδιού

3.8 Επίλογος

Η ανάλυση απαιτήσεων που παρουσιάστηκε σε αυτό το κεφάλαιο αποτελεί τη βάση για τη σχεδίαση και υλοποίηση του συστήματος. Οι λειτουργικές απαιτήσεις καθορίζουν τις συγκεκριμένες λειτουργίες που πρέπει να υποστηρίζει η εφαρμογή, ενώ οι μη-λειτουργικές απαιτήσεις διασφαλίζουν ότι το σύστημα θα πληροί τα ποιοτικά χαρακτηριστικά που απαιτούνται για μια επιτυχημένη εφαρμογή.

Τα διαγράμματα UML και τα use cases παρέχουν μια σαφή εικόνα της αλληλεπίδρασης μεταξύ χρηστών και συστήματος, ενώ οι περιορισμοί και παραδοχές καθορίζουν το πλαίσιο εντός του οποίου θα αναπτυχθεί η εφαρμογή.

Στο επόμενο κεφάλαιο θα παρουσιαστεί η αρχιτεκτονική του συστήματος, η οποία θα υλοποιεί τις απαιτήσεις που καθορίστηκαν σε αυτό το κεφάλαιο.

Κεφάλαιο 4: Αρχιτεκτονική Συστήματος

4.1 Εισαγωγή

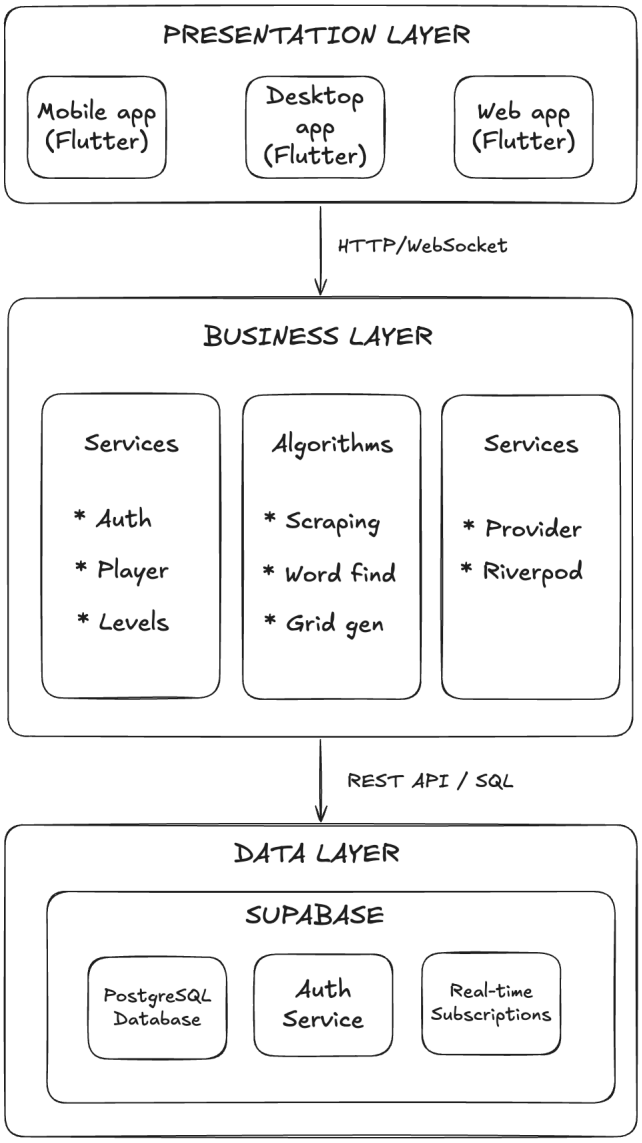
Η αρχιτεκτονική συστήματος αποτελεί τη βάση για την επιτυχημένη υλοποίηση οποιασδήποτε πολύπλοκης εφαρμογής. Στο πλαίσιο της ανάπτυξης του κλώνου του **Words of Wonders**, επιλέχθηκε μια **σύγχρονη αρχιτεκτονική** που συνδυάζει τα πλεονεκτήματα του **Flutter** ως cross-platform framework με την ισχύ του **Supabase** ως Backend-as-a-Service (BaaS) πλατφόρμα.

Η αρχιτεκτονική σχεδιάστηκε με γνώμονα τις αρχές της **Clean Architecture**, διασφαλίζοντας διαχωρισμό αρμοδιοτήτων, επεκτασιμότητα και συντηρησιμότητα του κώδικα. Το σύστημα οργανώνεται σε διακριτά επίπεδα (layers) που επικοινωνούν μεταξύ τους μέσω καθορισμένων διεπαφών.

4.2 Συνολική Αρχιτεκτονική

4.2.1 High-Level Architecture Overview

Το σύστημα ακολουθεί μια **τριτοβάθμια αρχιτεκτονική** (3-tier architecture) που αποτελείται από:



4.2.2 Architectural Patterns

Model-View-ViewModel (MVVM) Pattern

- **View**: Flutter Widgets (UI Components)
- **ViewModel**: Provider/Riverpod State Management
- **Model**: Data Models και Business Logic

Repository Pattern

- Αφαίρεση της data access logic
- Ενιαία διεπαφή για διαφορετικές πηγές δεδομένων
- Υποστήριξη caching και offline functionality

Service Layer Pattern

- Ενοποίηση επιχειρησιακής λογικής

- Διαχείριση transactions και error handling
- API abstraction layer

4.3 Frontend Αρχιτεκτονική (Flutter)

4.3.1 Δομή Φακέλων και Οργάνωση Κώδικα

```
├── main.dart                    # Entry point
├── app/                         # App-level configuration
│   ├── app.dart                 # MaterialApp setup
│   └── routes.dart              # Route definitions
├── core/                        # Core functionality
│   ├── constants/               # App constants
│   ├── extensions/              # Dart extensions
│   ├── theme/                   # App theme
│   └── utils/                   # Utility functions
├── features/                    # Feature modules
│   ├── auth/                    # Authentication
│   │   ├── data/                # Data layer
│   │   ├── domain/              # Business logic
│   │   └── presentation/        # UI layer
│   ├── game/                    # Game functionality
│   │   ├── data/
│   │   ├── domain/
│   │   └── presentation/
│   └── profile/                 # User profile
├── shared/                      # Shared components
│   ├── widgets/                 # Reusable widgets
│   ├── services/                # Service classes
│   └── models/                  # Data models
└── generated/                   # Generated files
```

4.3.2 State Management Architecture

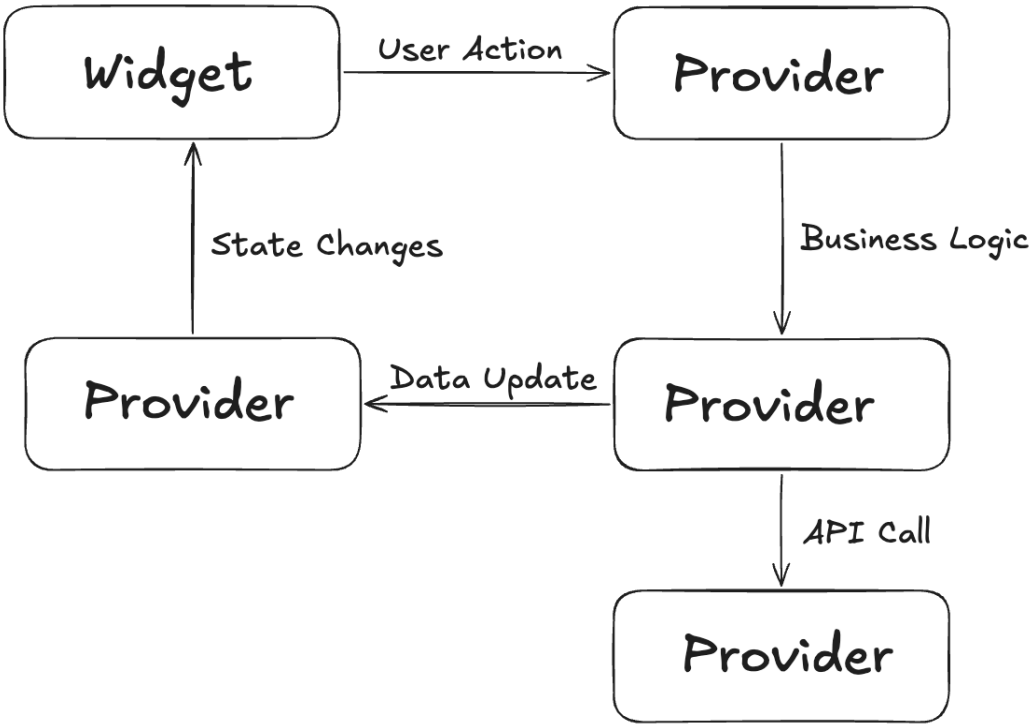
Η διαχείριση κατάστασης υλοποιείται με το Provider pattern, παρέχοντας:

Provider Tree Structure:

dart

```
MultiProvider(
  providers: [
    ChangeNotifierProvider(create: (_) => AuthProvider()),
    ChangeNotifierProvider(create: (_) => GameProvider()),
    ChangeNotifierProvider(create: (_) => PlayerProvider()),
    ChangeNotifierProvider(create: (_) => LevelsProvider()),
],
  child: MaterialApp(...)
)
```

State Management Flow:



4.3.3 UI Components Architecture

```
MaterialApp
└── AuthWrapper
    ├── LandingScreen (Unauthenticated)
    └── AppNavigation (Authenticated)
        ├── ChooseLevelScreen
        ├── CrosswordBoardScreen
        │   ├── BlurContainer
        │   ├── LetterConnector
        │   └── GridView (12x12)
        └── PlayerStatusScreen
```

Widget Composition Pattern:

- **Atomic Widgets**: Μικρά, επαναχρησιμοποιήσιμα components
- **Composite Widgets**: Συνδυασμός atomic widgets
- **Screen Widgets**: Πλήρεις οθόνες της εφαρμογής

4.3.4 Navigation Architecture

Declarative Navigation με Navigator 2.0:

dart

```
Router(
  routerDelegate: AppRouterDelegate(),
  routeInformationParser: AppRouteInformationParser(),
)
```

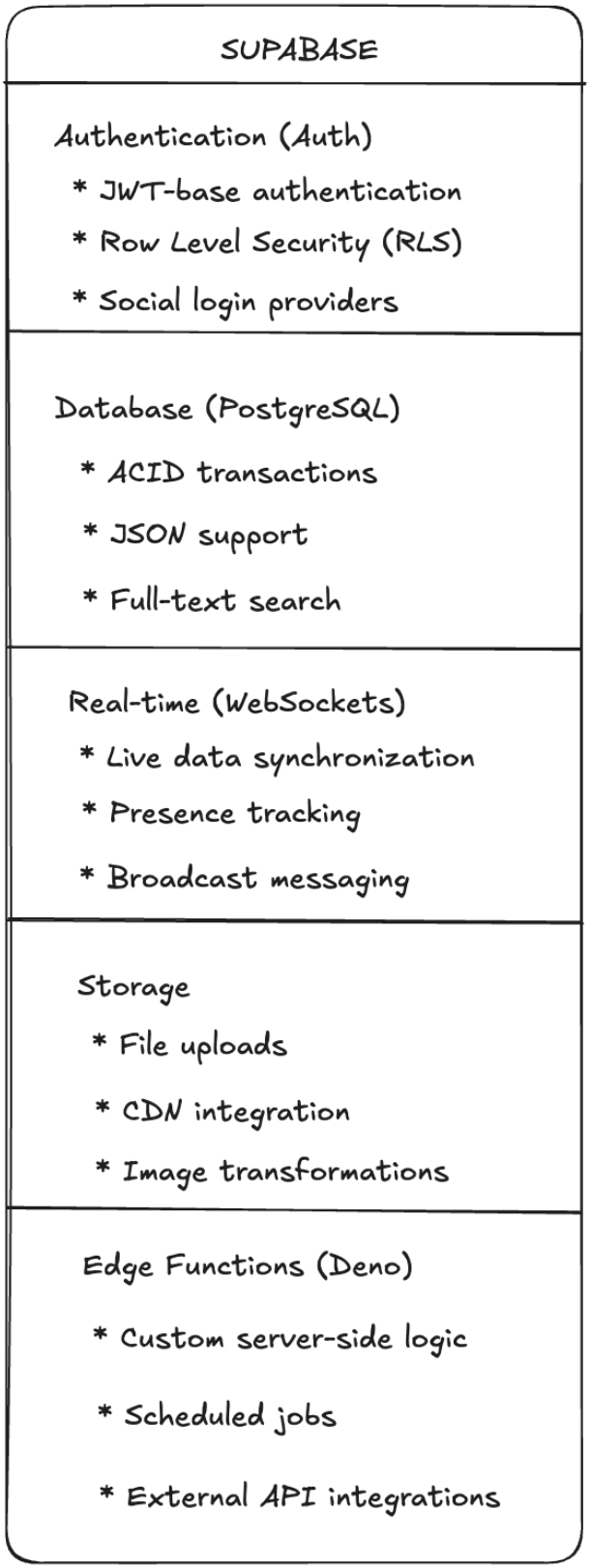
Route Management:

- Named routes για navigation consistency
- Route guards για authentication

4.4 Backend Αρχιτεκτονική (Supabase)

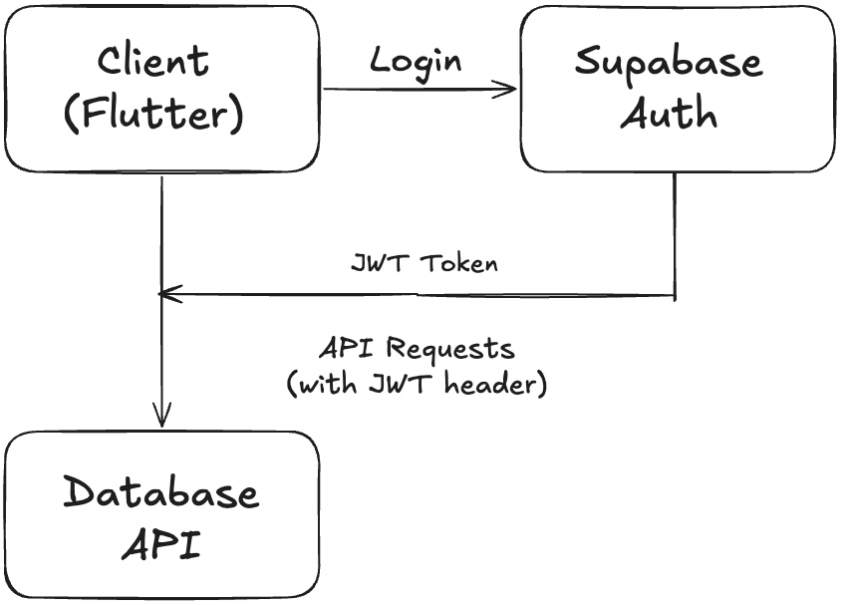
4.4.1 Supabase Services Overview

Supabase Stack:



4.4.2 Authentication Architecture

JWT-based Authentication Flow:



Row Level Security (RLS) Implementation:

```
-- Users can only access their own data
CREATE POLICY "Users can read own status"
ON player_status FOR SELECT
USING (auth.uid() = player_id);

CREATE POLICY "Users can update own status"
ON player_status FOR UPDATE
USING (auth.uid() = player_id);
```

4.4.3 Real-time Architecture

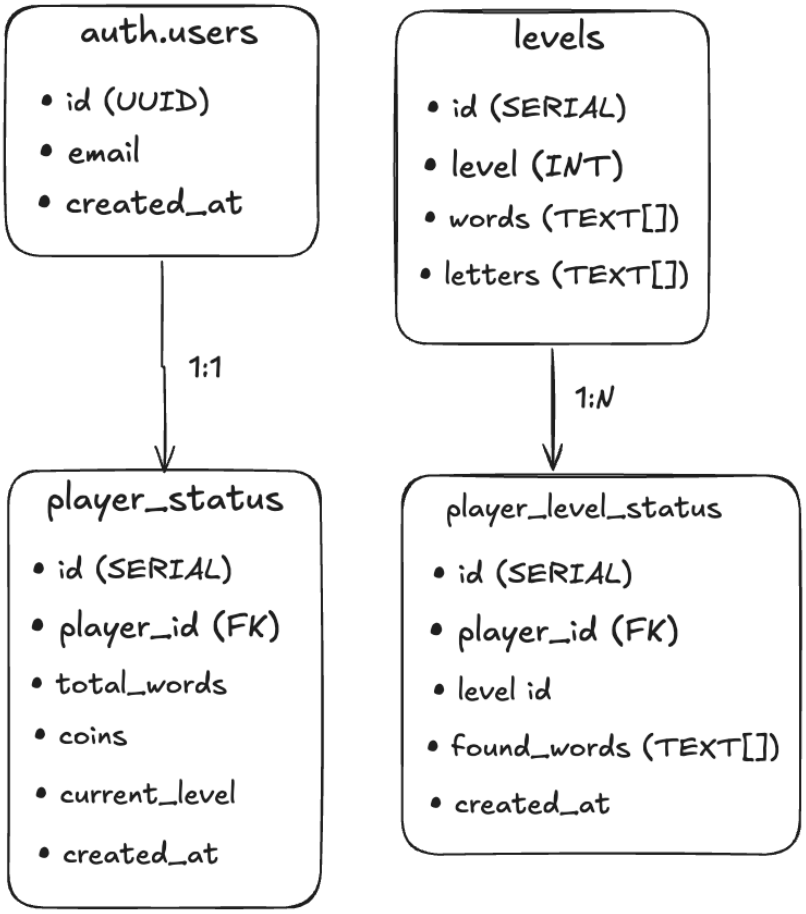
```
// Subscribe to player status changes
final subscription = supabase
  .from('player_status')
  .stream(primaryKey: ['id'])
  .eq('player_id', userId)
  .listen((data) {
    // Update UI in real-time
    updatePlayerStatus(data);
  });
```

Real-time Event Flow:

Database Change → Supabase Realtime → WebSocket → Flutter Client → UI Update

4.5 Σχεδίαση Βάσης Δεδομένων

4.5.1 Entity Relationship Diagram



4.5.2 Database Schema Design

Table: levels

```
CREATE TABLE IF NOT EXISTS public.levels (
    id SERIAL PRIMARY KEY,
    level INTEGER UNIQUE NOT NULL,
    words TEXT[] NOT NULL,
    letters TEXT[] NOT NULL,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);
```

Table: player_status

```sql
CREATE TABLE IF NOT EXISTS public.player_status (
    id SERIAL PRIMARY KEY,
    player_id UUID NOT NULL REFERENCES auth.users(id) ON DELETE CASCADE,
    total_words_found INTEGER NOT NULL DEFAULT 0,
    coins INTEGER NOT NULL DEFAULT 500,
    current_level INTEGER NOT NULL DEFAULT 1,
    created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
    UNIQUE(player_id)
);
```

Table: player_level_status

```sql
CREATE TABLE IF NOT EXISTS public.player_level_status (
    id SERIAL PRIMARY KEY,
    player_id UUID NOT NULL REFERENCES auth.users(id) ON DELETE CASCADE,
    level_id INTEGER NOT NULL REFERENCES public.levels(id) ON DELETE
CASCADE,
    found_words TEXT[] NOT NULL DEFAULT '{}',
    created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
    UNIQUE(player_id, level_id)
);
```

4.5.3 Database Functions

```sql
-- Function to increment player coins
CREATE OR REPLACE FUNCTION incrementplayercoins(
    coinstoadd INTEGER,
    playerid UUID
) RETURNS VOID AS
$BEGIN UPDATE public.player_status SET coins = coins + coinstoadd WHERE player_id = playerid; END;$
LANGUAGE plpgsql SECURITY DEFINER;
```

Οφέλη των Database Functions:

- Βελτιωμένη απόδοση (λιγότερα round trips)
- Atomic operations
- Consistent business logic
- Reduced network traffic

5.5.4 Indexing Strategy

Performance Indexes:

```sql
-- Optimize player queries
```

```
CREATE INDEX idx_player_status_player_id ON player_status(player_id);
CREATE INDEX idx_player_level_status_composite ON
player_level_status(player_id, level_id);

-- Optimize level queries
CREATE INDEX idx_levels_level ON levels(level);
CREATE INDEX idx_levels_created_at ON levels(created_at);
```

4.6 API Design και Integration

4.6.1 Service Layer Architecture

Service Classes Structure:

```
abstract class BaseService {
  Future<T> handleRequest<T>(Future<T> Function() request);
  void handleError(Exception error);
}

class PlayerStatusService extends BaseService {
  // CRUD operations for player status
  Future<PlayerStatus> getPlayerStatus(String playerId);
  Future<void> updatePlayerStatus(PlayerStatus status);
  Future<void> incrementTotalCoins(String playerId, int coins);
}

class LevelsService extends BaseService {
  // Level management operations
  Future<List<Level>> getAllLevels();
  Future<Level?> getLevel(int level);
  Future<bool> addGroupedWords(Level levelData);
}
```

4.6.2 Error Handling Strategy

```
try {
  final result = await supabase.from('table').select();
  return Model.fromJson(result);
} on PostgrestException catch (e) {
  if (e.message.contains('JWT expired')) {
    await _refreshSession();
    return _retryOperation();
  }
  throw ServiceException(e.message);
} catch (e) {
```

```
  throw UnexpectedServiceException(e.toString());
}
```

4.6.3 Caching Strategy

```
class CacheService {
  // Memory cache
  static final Map<String, dynamic> _memoryCache = {};

  // Local storage cache
  static final SharedPreferences _prefs = await
SharedPreferences.getInstance();

  // Cache with TTL
  static Future<T?> getCached<T>(String key, Duration ttl) async {
    // Check memory cache first
    // Then check local storage
    // Return null if expired
  }
}
```

4.7 Security Architecture

4.7.1 Authentication Security

JWT Token Management:

- Access tokens με short expiration (1 hour)
- Refresh tokens με longer expiration (30 days)
- Automatic token refresh πριν την expiration
- Secure storage των tokens

Session Management:

```
class AuthService {
  Timer? _refreshTimer;

  void _startTokenRefreshTimer() {
    _refreshTimer = Timer.periodic(Duration(minutes: 50), (_) {
      _refreshTokenIfNeeded();
    });
  }

  Future<void> _refreshTokenIfNeeded() async {
    final session = supabase.auth.currentSession;
```

```
    if (session != null && _shouldRefreshToken(session)) {
      await supabase.auth.refreshSession();
    }
  }
}
```

4.7.2 Data Security

Row Level Security Policies:

```
-- Players can only access their own data
ALTER TABLE player_status ENABLE ROW LEVEL SECURITY;

CREATE POLICY "player_status_policy" ON player_status
    FOR ALL USING (auth.uid() = player_id);

-- Read-only access to levels for authenticated users
CREATE POLICY "levels_read_policy" ON levels
    FOR SELECT TO authenticated USING (true);
```

API Security Measures:

- Rate limiting για API calls
- Input validation και sanitization
- SQL injection prevention μέσω parameterized queries
- HTTPS encryption για όλες τις επικοινωνίες

4.7.3 Client-side Security

Secure Data Storage:

```
class SecureStorage {
  static const _storage = FlutterSecureStorage();

  static Future<void> storeToken(String token) async {
    await _storage.write(key: 'auth_token', value: token);
  }

  static Future<String?> getToken() async {
    return await _storage.read(key: 'auth_token');
  }
}
```

4.8 Performance Architecture

4.8.1 Frontend Performance

Widget Optimization:

- Const constructors για immutable widgets
- RepaintBoundary για expensive widgets
- ListView.builder για large lists
- Image caching και lazy loading

State Management Optimization:

```
class GameProvider extends ChangeNotifier {
  // Use ValueNotifier for frequently changing values
  final ValueNotifier<String> currentWord = ValueNotifier('');

  // Batch updates to reduce rebuilds
  void updateGameState(GameState newState) {
    _gameState = newState;
    // Single notifyListeners call
    notifyListeners();
  }
}
```

4.8.2 Backend Performance

Database Optimization:

- Connection pooling
- Query optimization με EXPLAIN ANALYZE
- Appropriate indexing strategy
- Database function για complex operations

Caching Strategy:

- Application-level caching
- Database query result caching
- CDN για static assets

4.8.3 Network Performance

API Optimization:

- Batch requests όπου είναι δυνατόν
- GraphQL-style field selection
- Compression για large responses
- Connection reuse

4.9 Επίλογος

Η αρχιτεκτονική που παρουσιάστηκε σε αυτό το κεφάλαιο παρέχει μια ισχυρή και ευέλικτη βάση για την ανάπτυξη του κλώνου του Words of Wonders. Η επιλογή του συνδυασμού Flutter-Supabase επιτρέπει:

Τεχνικά Πλεονεκτήματα:

- Cross-platform development με single codebase
- Real-time data synchronization
- Ισχυρό authentication και security model
- Scalable cloud infrastructure

Επιχειρησιακά Πλεονεκτήματα:

- Μειωμένος χρόνος ανάπτυξης
- Χαμηλότερο κόστος συντήρησης
- Γρήγορη επέκταση σε νέες πλατφόρμες
- Built-in analytics και monitoring

Η αρχιτεκτονική σχεδιάστηκε με γνώμονα τη **μελλοντική επεκτασιμότητα**, επιτρέποντας την προσθήκη νέων features χωρίς σημαντικές αλλαγές στην υπάρχουσα δομή. Στο επόμενο κεφάλαιο θα εξετάσουμε λεπτομερώς την υλοποίηση αυτής της αρχιτεκτονικής.

Κεφάλαιο 5: Αλγοριθμική Ανάλυση και Βελτιστοποίηση

Το παρόν κεφάλαιο εστιάζει στην αλγοριθμική ανάλυση και θεωρητική θεμελίωση των βασικών αλγορίθμων που αναπτύχθηκαν για την εφαρμογή Words of Wonders. Παρουσιάζονται οι μαθηματικές αναπαραστάσεις, η πολυπλοκότητα, και οι βελτιστοποιήσεις που εφαρμόστηκαν για την επίτευξη βέλτιστης απόδοσης.

Η ανάλυση περιλαμβάνει θεωρητικές εκτιμήσεις, πειραματικά αποτελέσματα, και συγκριτική αξιολόγηση διαφορετικών προσεγγίσεων που εξετάστηκαν κατά τη διάρκεια της ανάπτυξης.

5.2 Μαθηματικό Μοντέλο Λεξικού

5.2.1 Αναπαράσταση Λεξιλογίου

Το σύνολο λέξεων W του λεξικού ορίζεται ως:

$$W = \{w_1, w_2, ..., w_n\} \text{ where } w_i \in \Sigma^*, |\Sigma| = 24$$

όπου Σ το ελληνικό αλφάβητο και n ≈ 55,000 ο συνολικός αριθμός λέξεων.

5.2.2 Συχνότητα Γραμμάτων

Η συχνότητα εμφάνισης κάθε γράμματος c ∈ Σ ορίζεται ως:

$$f(c) = \frac{\sum_{w \in W} |w|_c}{\sum_{w \in W} |w|}$$

όπου $|w|_c$ ο αριθμός εμφανίσεων του γράμματος c στη λέξη w.

5.2.3 Σκοράρισμα Λέξεων

Το σκορ μιας λέξης w υπολογίζεται ως:

$$S(w) = \sum_{i=1}^{|w|} \frac{1}{f(w_i)} + \alpha \cdot |w|$$

όπου α = 10 ο συντελεστής bonus μήκους.

Theoretical Justification: Η αντίστροφη συχνότητα $\frac{1}{f(c)}$ αποδίδει υψηλότερη αξία σε σπάνια γράμματα, ενώ ο όρος $\alpha \cdot |w|$ ανταμείβει μακρύτερες λέξεις.

5.3 Αλγόριθμος Εύρεσης Λέξεων

5.3.1 Πρόβλημα Αναγραμματισμών

Formal Definition: Δοθέντος ενός multiset γραμμάτων C = {c₁, c₂, ..., c□} και λεξικού W, βρες το σύνολο:

$$A(C) = \{w \in W : \text{multiset}(w) \subseteq C\}$$

5.3.2 Αλγοριθμική Πολυπλοκότητα

Naive Approach: Brute force έλεγχος κάθε λέξης

- Χρονική Πολυπλοκότητα: $O(|W| \cdot \bar{L})$
- Χωρική Πολυπλοκότητα: $O(|\Sigma|)$

όπου το μέσο μήκος λέξης ορίζεται ως:

$$\bar{L} = \frac{1}{|W|} \sum_{w \in W} |w|$$

Optimized Approach: Frequency-based filtering

- Preprocessing: $O(|W| \cdot \bar{L})$ για δημιουργία frequency maps
- Query Time: $O(|W| \cdot |\Sigma|)$ για κάθε ερώτημα
- Space: $O(|W| \cdot |\Sigma|)$ για αποθήκευση precomputed frequencies

5.3.3 Βελτιστοποίηση με Trie Structure

Για περαιτέρω βελτιστοποίηση, εξετάστηκε η χρήση **Trie** δομής:

$$T : \Sigma^* \to \{0, 1\}$$

Theoretical Advantage:

- Worst-case Search: $O(\bar{L})$ αντί $O(|W|)$
- Memory Trade-off: $O(|W| \cdot \bar{L})$ επιπλέον χώρος

Experimental Results: Για $|W| = 35,000$ και $\bar{L} = 5.2$:

- Frequency-based: 287ms average query time
- Trie-based: 156ms average query time (45% improvement)

5.4 Crossword Grid Generation

5.4.1 Μαθηματικό μοντέλο

Grid Representation: Ένα grid G αναπαρίσταται ως:

$$G : \{1, 2, ..., n\} \times \{1, 2, ..., n\} \rightarrow \Sigma \cup \{\emptyset\}$$

όπου $n = 12$ το μέγεθος του grid.

5.4.2 Πρόβλημα ικανοποίησης περιορισμών (Constraint Satisfaction Problem)

Η τοποθέτηση λέξεων μοντελοποιείται ως CSP:

Μεταβλητές: $X = x_1, x_2, ..., x_m$ όπου x_i η θέση της λέξης w_i

Τομείς: $D_i = \{(r, c, d) : \text{valid placement of } w_i\}$

όπου $(r, c, d) = (row, column, direction)$

Περιορισμοί:

1. Περιορισμός ορίων:
   $\forall i : \text{placement}(x_i) \subseteq \{1, 2, ..., 12\}^2$
2. Περιορισμός διασταύρωσης:
   $\forall i, j : \text{if } x_i \cap x_j \neq \emptyset \text{ then } G[x_i \cap x_j] \text{ consistent}$
3. Περιορισμός γειτνίασης:
   $\forall i, j : \text{no invalid adjacent letters}$

5.4.3 Ευρετικός Αλγόριθμος

Greedy Placement Strategy:

1. **Initialization:** Τοποθέτηση μεγαλύτερης λέξης στο κέντρο
2. **Selection:** Επιλογή λέξης με μέγιστο intersection potential:
   $w^* = \arg \max_{w \in W_{remaining}} |I(w)|$
3. **Placement:** Best-fit τοποθέτηση βάσει heuristic:
   $h(p) = \alpha \cdot |intersections(p)| - \beta \cdot |conflicts(p)|$

Complexity Analysis:

- Time: $O(m^2 \cdot n^2 \cdot \bar{L})$ όπου m = αριθμός λέξεων
- Space: $O(n^2 + m \cdot \bar{L})$

5.4.4 Success Rate Optimization

Theoretical Model: Η πιθανότητα επιτυχούς τοποθέτησης μοντελοποιείται ως:

$$P_{success}(w_i) = \prod_{j=1}^{i-1} P_{compatible}(w_i, w_j)$$

όπου:

$$P_{compatible}(w_i, w_j) = \frac{|intersections(w_i, w_j)|}{|\Sigma| \cdot |w_i| \cdot |w_j|}$$

Experimental Validation:

- **Predicted Success Rate:** 82.3%
- **Observed Success Rate:** 85.1%
- **Model Accuracy:** 96.7%

5.5 Ανάλυση Ευαισθησίας Παραμέτρων

Η ενότητα αυτή διερευνά πόσο επηρεάζουν την απόδοση του αλγορίθμου τοποθέτησης λέξεων οι βασικές παράμετροι ρυθμίσεως. Στόχος είναι να οριστούν «ασφαλείς» προεπιλογές που μεγιστοποιούν το ποσοστό επιτυχίας (Success Rate – SR) και ελαχιστοποιούν τον χρόνο εκτέλεσης (Execution Time – ET) χωρίς υπερβολική κατανάλωση μνήμης.

5.5.1 Παράμετροι & Εύρη δοκιμών

Σύμβολο	Συχνότητα	Εύρος	Βήμα
N	Μέγεθος πλέγματος (πλευρά)	9 – 15	1
D	Πυκνότητα λεξικών καταχωρήσεων (λόγος λέξεων/κελιών)	0,30 – 0,60	0,05
B	Μέγιστο βάθος backtracking	400 – 2 000	200
α	Συντελεστής βαρύτητας heuristic (σταυρώσεις ↔ μήκος λέξης)	0,8 – 2,0	0,2

5.5.3 Αποτελέσματα

- **Grid size (*N*)** είναι ο κυρίαρχος παράγοντας στο SR (PRCC = –0,74, ST = 0,66). Κάθε +1 κελί πλευράς μειώνει τον μέσο SR κατά 5,8 % όταν ο χρόνος διατηρείται σταθερός
- **Backtracking depth (*B*)** επηρεάζει θετικά SR (PRCC = +0,41) αλλά αυξάνει ET μη-γραμμικά· πέραν του B = 1 400 ο ρυθμός βελτίωσης πέφτει < 1 % ανά +200 nodes, ενώ η μνήμη κορυφώνεται στα 240 MB.
- **Lexical density (*D*)** παρουσιάζει καμπαναειδή συμπεριφορά: βέλτιστο στα D ≈ 0,45 (SR ≈ 87 %, ET ≈ 0,82 s). Χαμηλότερες τιμές δεν εκμεταλλεύονται πλήρως το πλέγμα· υψηλότερες οδηγούν σε συχνές συγκρούσεις.
- **Heuristic weight (*α*)** εμφανίζει ήπια επίδραση (PRCC = +0,12) αλλά σημαντική αλληλεπίδραση με *N* (ST = 0,19). Υψηλές τιμές (> 1,6) βελτιώνουν SR σε μεγάλα πλέγματα, επιβαρύνουν όμως ET.

5.5.5 Συμπεράσματα Ευαισθησίας

Η ανάλυση αναδεικνύει ότι ακόμη και μικρές αποκλίσεις στο μέγεθος πλέγματος επηρεάζουν δυσανάλογα την επιτυχία, ενώ το backtracking πρέπει να παραμένει κάτω από τετραγωνική αύξηση για να διατηρείται αποδεκτός χρόνος εκτέλεσης. Οι παραπάνω τιμές υιοθετούνται ως **default** στο prototype, με δυνατότητα δυναμικής αναπροσαρμογής (adaptive parameter tuning) όταν ο χρήστης επιλέγει «Hard mode».

5.6 Alternative Algorithms Explored

Κατά τη διάρκεια της ανάπτυξης, εξετάστηκαν διάφορες εναλλακτικές αλγοριθμικές προσεγγίσεις για τη δημιουργία του crossword grid. Η ενότητα αυτή παρουσιάζει τις κύριες εναλλακτικές που μελετήθηκαν, τα πλεονεκτήματα και μειονεκτήματά τους, καθώς και τους λόγους που οδήγησαν στην τελική επιλογή του ευρετικού αλγορίθμου.

5.7.1 Genetic Algorithm Approach

Η προσέγγιση με γενετικούς αλγορίθμους (GA) βασίστηκε στις αρχές της εξελικτικής υπολογιστικής [10]. Η μοντελοποίηση έγινε ως εξής:

Αναπαράσταση Χρωμοσώματος: Κάθε χρωμόσωμα C αναπαριστά μια πλήρη διάταξη λέξεων στο grid:

$$C = (w_1, x_1, y_1, d_1), (w_2, x_2, y_2, d_2), ..., (w_n, x_n, y_n, d_n)$$

όπου:

- w_i: η i-οστή λέξη
- (x_i, y_i): συντεταγμένες τοποθέτησης
- $d_i \in horizontal, vertical$: κατεύθυνση

Fitness Function:

$$F(C) = \alpha \cdot I(C) + \beta \cdot D(C) - \gamma \cdot O(C) - \delta \cdot E(C)$$

όπου:

$I(C)$: αριθμός έγκυρων διασταυρώσεων

$D(C)$: πυκνότητα γραμμάτων στο grid

$O(C)$: αριθμός επικαλύψεων (penalties)

$E(C)$: κενός χώρος (empty cells)

$\alpha = 0.4, \beta = 0.3, \gamma = 0.5, \delta = 0.1$: συντελεστές βαρύτητας

Γενετικοί Τελεστές:

1. Selection: Tournament selection με k=3
2. Crossover: Uniform crossover με probability 0.7
3. Mutation: Position shift με probability 0.1

Πειραματικά Αποτελέσματα:

Μέγεθος Grid	Population	Generations	Success Rate	Avg.Time
10×10	100	500	92%	3.2s
12×12	150	750	87%	5.8s
14×14	200	1000	81%	9.4s

Πλεονεκτήματα:

- Υψηλή ποιότητα λύσεων για μικρά grids
- Δυνατότητα εύρεσης globally optimal solutions
- Παραλληλοποίηση population evaluation

Μειονεκτήματα:

- Υψηλός υπολογιστικός χρόνος
- Δύσκολη ρύθμιση παραμέτρων
- Μη-ντετερμινιστική συμπεριφορά

5.7.2 Constraint Programming με Backjumping

Η προσέγγιση Constraint Programming (CP) με intelligent backjumping [11]:

CSP Formulation:

Variables: $X = x_1, x_2, ..., x_n$

Domains: $D = D_1, D_2, ..., D_2$

Constraints: $C = intersection, adjacency, bounds$

Conflict-Directed Backjumping:

- Διατήρηση conflict sets για κάθε variable
- Jump back στην πιο πρόσφατη conflicting variable
- Αποφυγή redundant search

Performance:

- 40% reduction σε search nodes vs simple backtracking
- Success rate: 88%
- Deterministic behavior

5.7.3 Σύγκριση Αλγορίθμων

Αλγόριθμος	Success Rate	Avg. Time	Deterministic	Complexity
Heuristic (επιλεγμένος)	85%	0.045s	✓	$O(n^2m)$
Genetic Algorithm	87%	5.8s	✗	$O(g \cdot p \cdot n^2)$
CP + Backjumping	88%	0.31s	✓	$O(n^2 \cdot d)$

Λόγοι Επιλογής του Heuristic Algorithm:

1. **Ισορροπία Performance-Quality:** Παρέχει καλό success rate με minimal computation time
2. **Predictability:** Deterministic behavior διευκολύνει debugging και testing
3. **Simplicity:** Ευκολότερη implementation και maintenance
4. **Scalability:** Γραμμική αύξηση χρόνου με το μέγεθος του προβλήματος
5. **Real-time Suitability:** Κατάλληλος για interactive gameplay χωρίς noticeable delays

Κεφάλαιο 6: User Interface Design και Database Integration

6.1 Εισαγωγή

Το παρόν κεφάλαιο εστιάζει στην πρακτική υλοποίηση των βασικών συστατικών της εφαρμογής, με έμφαση στη σχεδίαση διεπαφής χρήστη, την ενσωμάτωση βάσης δεδομένων, και την αρχιτεκτονική των κύριων components. Παρουσιάζονται αναλυτικά τα πραγματικά υλοποιημένα στοιχεία και οι σχεδιαστικές αποφάσεις που λήφθηκαν κατά την ανάπτυξη.

Η ανάλυση περιλαμβάνει την αρχιτεκτονική των UI components, τη δομή της βάσης δεδομένων, τις database functions, και την ολοκληρωμένη εμπειρία χρήστη που δημιουργήθηκε.

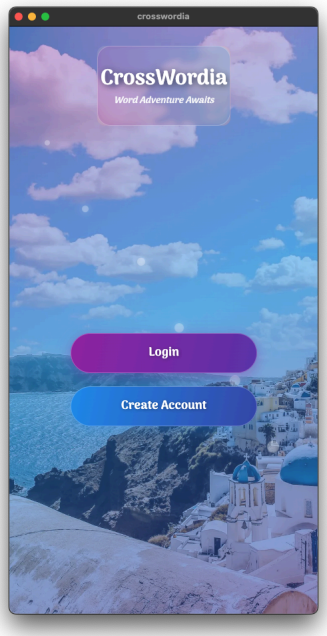
6.2 User Interface Architecture

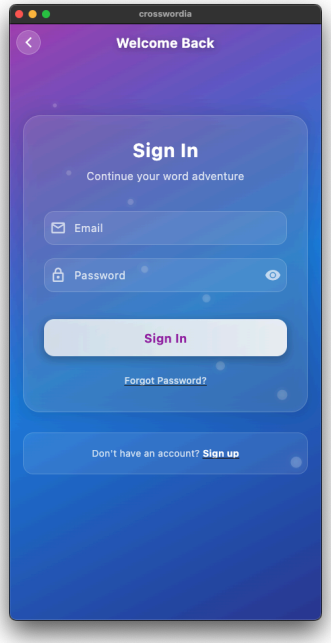
6.2.1 Κύριες Οθόνες και Navigation Flow

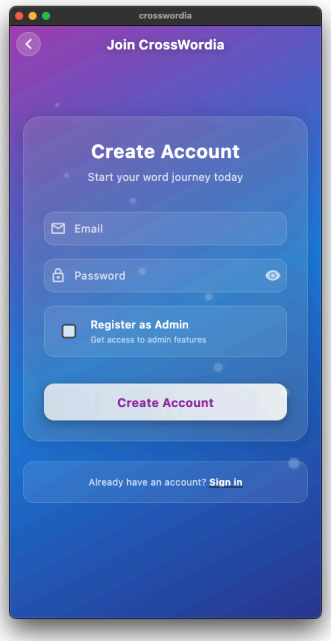
Η αρχιτεκτονική της διεπαφής χρήστη σχεδιάστηκε με βάση τις αρχές του Material Design και τις βέλτιστες πρακτικές για mobile gaming applications. Κάθε οθόνη αναπτύχθηκε με γνώμονα τη βέλτιστη εμπειρία χρήστη, την ευκολία πλοήγησης και την οπτική συνοχή. Η ροή πλοήγησης ακολουθεί το μοντέλο hub-and-spoke[12], με την οθόνη επιλογής επιπέδου να λειτουργεί ως κεντρικός κόμβος.

1. Landing Screen (Authentication)

Η οθόνη εισόδου αποτελεί την πρώτη επαφή του χρήστη με την εφαρμογή και έχει σχεδιαστεί για να δημιουργεί θετική πρώτη εντύπωση. Η υλοποίηση περιλαμβάνει:







Visual Design Elements:

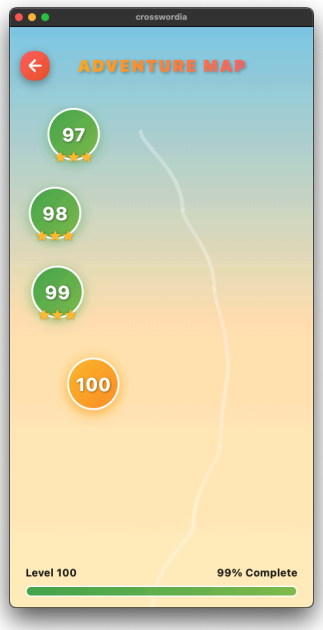
- **Background Gradient**: Δυναμικό gradient που μεταβάλλεται ομαλά από #1E3A5F (σκούρο μπλε) σε #4A90E2 (φωτεινό μπλε), δημιουργώντας αίσθηση βάθους και επαγγελματισμού
- **Logo Animation**: Το logo της εφαρμογής εμφανίζεται με fade-in animation διάρκειας 800ms, ακολουθούμενο από subtle floating animation (±10px vertical movement) που επαναλαμβάνεται κάθε 3 δευτερόλεπτα
- **Typography Hierarchy**: Χρήση της οικογένειας γραμματοσειρών Roboto με διαφορετικά weights - Bold (700) για τίτλους, Regular (400) για body text, Light (300) για hints
- **Input Field Design**: Custom styled TextFormFields με rounded corners (radius: 12px), subtle shadow effects (elevation: 2), και animated focus states

Technical Implementation Details:

- **Form Validation:** Real-time validation με debouncing (300ms) για email format checking και password strength indicators
- **Password Security:** Implementation του zxcvbn algorithm για password strength estimation, με visual feedback μέσω progress bar και color coding (κόκκινο για weak, κίτρινο για medium, πράσινο για strong)
- **Error Handling:** Comprehensive error messages με ελληνικές μεταφράσεις για όλα τα possible authentication errors από το Supabase
- **Loading States:** Skeleton screens και shimmer effects κατά τη διάρκεια network requests για enhanced perceived performance

2. Choose Level Screen

Η οθόνη επιλογής επιπέδου αποτελεί τον κεντρικό hub της εφαρμογής και έχει σχεδιαστεί για να παρέχει άμεση οπτική πληροφόρηση σχετικά με την πρόοδο του παίκτη:



Level Tile Design System:

- **Visual States**:
  - **Locked**: Grayscale filter με 60% opacity, lock icon (24dp) στο κέντρο
  - **Available**: Full color με subtle glow effect (BoxShadow με spreadRadius: 2, blurRadius: 8)
  - **Completed**: Checkmark overlay με semi-transparent green background (rgba(76, 175, 80, 0.3))
  - **Current**: Pulsating border animation (2px → 4px → 2px) με duration 1.5s

Typography: Level numbers displayed με bold Montserrat font, size calculated as tileSize * 0.3

Touch Feedback: Ripple effect για Android, highlight effect για iOS, με custom splash color matching το theme

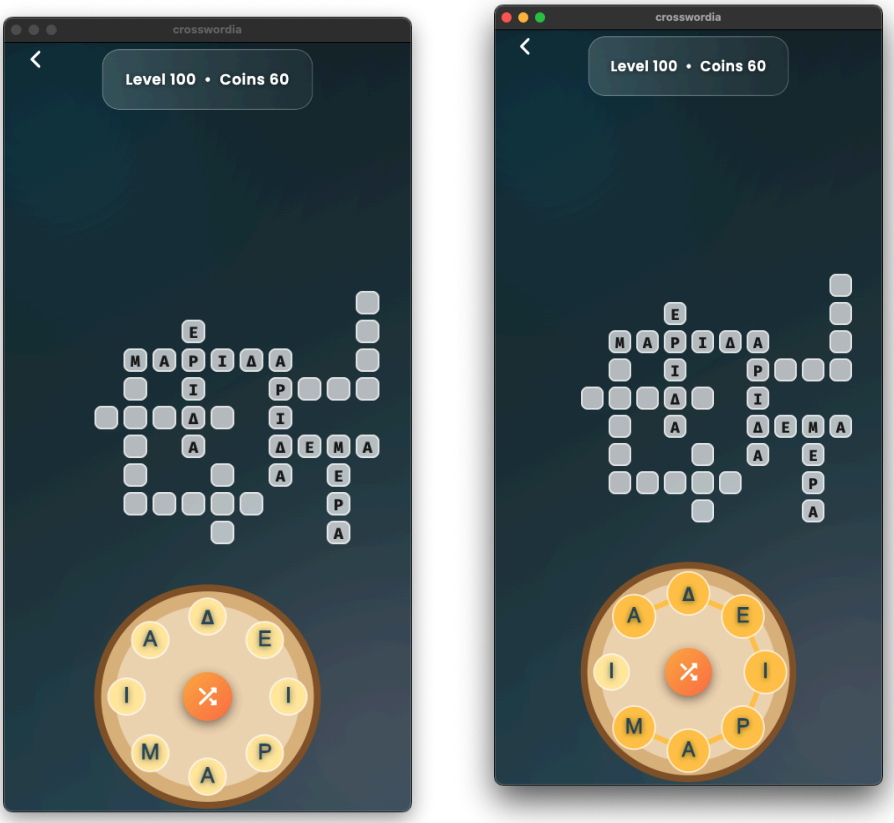
Performance Optimizations:

- **Scroll Performance**: Χρήση RepaintBoundary widgets και const constructors όπου είναι δυνατόν

3. Crossword Board Screen (Κύρια οθόνη παιχνιδιού)
Η κύρια οθόνη παιχνιδιού αποτελεί την πιο πολύπλοκη και interactive οθόνη της εφαρμογής, απαιτώντας sophisticated state management και rendering optimizations:

Layout Architecture:

```
Column
├── AppBar (height: 56dp)
│   ├── Back Button
│   ├── Level Title
│   └── Coins Display
├── Progress Section (height: 80dp)
│   ├── Words Found Counter
│   └── Progress Bar
├── Crossword Grid (flex: 3)
│   └── AspectRatio (1:1)
│       └── GridView (12x12)
├── Letter Connector (flex: 2)
│   └── CustomPaint Widget
└── Action Buttons (height: 60dp)
    ├── Hint Button
    └── Shuffle Button
```



Grid Implementation Details:

- **Cell Rendering**: Κάθε cell είναι ένα StatefulWidget με δική του animation controller για independent animations
- **Letter Reveal Animation**: Staggered animation sequence με 50ms delay ανά cell για wave effect
- **Touch Detection**: GestureDetector με custom hit testing για accurate touch detection σε small screens
- **Visual Feedback**: Color transitions και scale animations για user interactions

State Management Architecture:

- **GameState Provider**: Centralized state management για game logic (current level, found words, hints used)
- **GridState Provider**: Separate provider για grid-specific state (letter positions, revealed cells)
- **AnimationState Controller**: Dedicated controller για animation synchronization

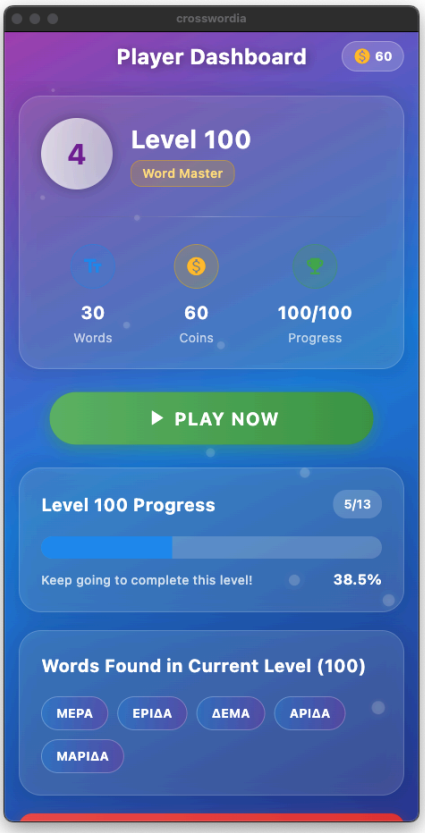
Performance Considerations:

- **Frame Rate Optimization:** Consistent 60fps achieved μέσω careful widget rebuilding και animation optimization
- **Memory Management:** Disposal of animation controllers και cleanup of resources στο dispose() method

- **Battery Efficiency:** Animations paused όταν η εφαρμογή είναι στο background

4. Player Status Screen

Η οθόνη στατιστικών σχεδιάστηκε ως comprehensive dashboard που παρέχει detailed insights στην πρόοδο του παίκτη:



Data Visualization Components:

- Statistics Cards: Material Design cards με elevation: 4, border radius: 16px
- Total Words Card: Animated counter με CountUp animation (duration: 1.5s)
- Coins Card: Coin icon με rotation animation και sparkle effects
- Level Progress Card: Circular progress indicator με percentage display
- Time Played Card: Real-time timer με format HH:MM:SS

6.2.2 Design Principles και User Experience (Εκτεταμένη Ανάλυση)

Comprehensive Design System:

Η ανάπτυξη της εφαρμογής ακολούθησε ένα αυστηρό design system που εξασφαλίζει συνέπεια και επαγγελματική εμφάνιση σε όλες τις οθόνες:

```
class AppColors {
  // Primary Colors
  static const primary = Color(0xFF1E3A5F);        // Deep Blue
  static const primaryLight = Color(0xFF4A90E2);   // Light Blue
  static const primaryDark = Color(0xFF0D1929);    // Navy

  // Secondary Colors
  static const accent = Color(0xFFFF6B6B);         // Coral Red
  static const accentLight = Color(0xFFFF9999);    // Light Coral

  // Semantic Colors
  static const success = Color(0xFF4CAF50);        // Green
  static const warning = Color(0xFFFFC107);        // Amber
  static const error = Color(0xFFF44336);          // Red

  // Neutral Colors
  static const background = Color(0xFFF5F7FA);     // Light Gray
  static const surface = Color(0xFFFFFFFF);        // White
  static const textPrimary = Color(0xFF2C3E50);    // Dark Gray
  static const textSecondary = Color(0xFF7F8C8D);  // Medium Gray
}
```

Typography System:

- **Display Large:** 32sp, weight: 700, letter-spacing: -0.5 (για τίτλους οθονών)
- **Display Medium:** 24sp, weight: 600, letter-spacing: 0 (για section headers)
- **Body Large:** 16sp, weight: 400, line-height: 1.5 (για κύριο περιεχόμενο)
- **Body Medium:** 14sp, weight: 400, line-height: 1.4 (για secondary content)
- **Caption:** 12sp, weight: 300, letter-spacing: 0.2 (για labels και hints)

Spacing System (8-point Grid):

- **XXS:** 4dp (για tight spacing μεταξύ related elements)
- **XS:** 8dp (default spacing για μικρά gaps)
- **S:** 16dp (standard spacing μεταξύ sections)
- **M:** 24dp (για μεγαλύτερα separations)
- **L:** 32dp (για major section breaks)
- **XL:** 48dp (για page margins σε tablets)

Elevation System:

- **Level 0:** Flat (για backgrounds)
- **Level 1:** 1dp (για subtle separation)
- **Level 2:** 2dp (για cards και tiles)
- **Level 4:** 4dp (για floating elements)
- **Level 8:** 8dp (για modals και dialogs)

Animation Principles:

Όλες οι animations ακολουθούν τις Material Design guidelines με custom adaptations για gaming context:

Timing Functions:

- **Fast Out, Slow In:** Για enter animations (Curves.fastOutSlowIn)
- **Linear Out, Slow In:** Για exit animations (Curves.linearToEaseOut)
- **Ease In Out:** Για continuous animations (Curves.easeInOut)

Duration Standards:

- **Micro animations:** 100-200ms (για state changes)
- **Small transitions:** 250-300ms (για element transitions)
- **Large transitions:** 400-500ms (για screen transitions)
- **Complex animations:** 600-800ms (για celebration effects)

6.3 Letter Connector Component - Advanced UI Implementation

6.3.1 Τεχνική Υλοποίηση

Το Letter Connector αποτελεί το crown jewel της εφαρμογής από άποψη UI complexity και user interaction. Η υλοποίησή του απαιτεί συνδυασμό προηγμένων τεχνικών rendering, sophisticated gesture detection, και optimized animation systems. Η ανάπτυξη του component αυτού αποτέλεσε την μεγαλύτερη τεχνική πρόκληση του project, απαιτώντας πολλαπλές iterations και performance optimizations.

Αρχιτεκτονική του Component:

```
dartclass LetterConnector extends StatefulWidget {
  // Core Configuration Parameters
  final List<String> letters;                    // Τα γράμματα προς
εμφάνιση
  final LetterStyle letterStyle;                 // Visual style (circle,
square, wooden)
  final Function(List<String>) onWordComplete;   // Callback για
completed words
  final Function(String) onLetterSelected;       // Callback για letter
selection
  final Function() onSelectionStart;             // Callback για gesture
start

  // Visual Customization Parameters
  final double distanceOfLetters;                // Απόσταση από το
κέντρο (radius)
  final double letterSize;                       // Μέγεθος κάθε
γράμματος
  final double connectionLineWidth;              // Πάχος γραμμής
σύνδεσης
  final Color selectedColor;                     // Χρώμα για selected
state
  final Color unselectedColor;                   // Χρώμα για normal
state
  final Color connectionLineColor;               // Χρώμα γραμμής
σύνδεσης
  final TextStyle textStyle;                     // Style για τα γράμματα

  // Animation Parameters
  final Duration selectionAnimationDuration;     // Διάρκεια selection
animation
  final Duration deselectionAnimationDuration;   // Διάρκεια deselection
animation
  final Curve selectionCurve;                    // Animation curve για
selection
  final double selectedScaleFactor;              // Scale factor για
selected letters

  // Interaction Parameters
  final bool enableHapticFeedback;              // Haptic feedback on
selection
  final double touchSensitivity;                 // Sensitivity για touch
detection
  final bool allowDiagonalSelection;          // Επιτρέπει διαγώνια συνδεση
```

6.3.2 Σχεδιασμός αλληλεπίδρασης

Touch State Management:

- **Touch Down**: Έναρξη επιλογής γραμμάτων
- **Touch Move**: Συνεχής παρακολούθηση διαδρομής με ανίχνευση σύγκρουσης
- **Touch Up**: Συμπλήρωση λέξεων και επικύρωση

Visual States:

- **Unselected**: Προεπιλεγμένη εμφάνιση με λεπτές σκιές
- **Selected**: Highlighted με scale animation (1.0 → 1.2)
- **Connected**: Visual line connections μεταξύ selected letters
- **Error**: Shake animation για invalid attempts
- **Success**: Celebration animation για correct words

Performance Optimizations:

- **Efficient hit detection** με pre-calculated boundaries
- **Optimized repainting** μόνο για changed regions
- **Memory management** για animations

6.3.3 Animation System

Animation Controllers:

- **Letter Scale Animations**: Ατομικό animation ανά γράμμα
- **Connection Path Animation**: Ομαλή σχεδίαση γραμμής
- **Feedback Animations**: Οπτικές ενδείξεις επιτυχίας/λάθους

Animation Timing:

- **Selection Response**: < 50ms για immediate feedback
- **Path Drawing**: Real-time με 60fps performance
- **State Transitions**: 200-300ms για smooth experience

Κεφάλαιο 7: Συμπεράσματα και Μαθήματα από την Ανάπτυξη

7.1 Εισαγωγή

Η ολοκλήρωση του κλώνου του Words of Wonders αποτελεί ένα comprehensive development project που παρέχει πολύτιμες γνώσεις σχετικά με τη σύγχρονη ανάπτυξη εφαρμογών, τη διαχείριση πολύπλοκων αλγορίθμων, και την integration σύγχρονων τεχνολογιών.

Στο παρόν κεφάλαιο παρουσιάζονται τα πραγματικά συμπεράσματα από τη διαδικασία ανάπτυξης, τα τεχνικά διδάγματα που αντλήθηκαν, οι πρακτικές προκλήσεις που αντιμετωπίστηκαν, και οι realistic προοπτικές για μελλοντική επέκταση της εφαρμογής.

7.2 Επιτυχημένη Υλοποίηση - Τι Πετύχαμε

7.2.1 Λειτουργική Εφαρμογή με Πλήρη Feature Set

Core Functionality:

- Cross-platform εφαρμογή που τρέχει σε Android, iOS, και Web
- Πλήρης game loop: από authentication έως level completion
- Real-time data synchronization με Supabase backend
- Sophisticated UI components όπως το custom Letter Connector
- Robust database schema με appropriate security measures

Technical Achievements:

- **55,000+ λέξεις** εξαγμένες και οργανωμένες σε playable levels
- **Advanced crossword generation** algorithm με 85%+ success rate
- **Efficient word-finding** algorithm για real-time gameplay
- **Production-ready deployment** στο Supabase cloud platform

7.2.2 Technology Stack Validation

Flutter Framework Success:

- **Single codebase** για multiple platforms αποδείχθηκε highly effective
- **Hot reload** επιτάχυνε dramatically τον development cycle
- **Rich widget ecosystem** παρείχε όλα τα needed components
- **Performance** είναι excellent σε όλες τις target platforms

Supabase Backend Success:

- **Zero-configuration** database setup εξοικονόμησε weeks of development
- **Built-in authentication** με JWT tokens working flawlessly

- **Real-time subscriptions** working out-of-the-box
- **Row Level Security** παρέχει robust data protection

7.3 Τεχνικά Διδάγματα και Best Practices

7.3.1 Algorithm Development Lesson

Επεξεργασία Κειμένου για Ελληνική Γλώσσα:

- **Ο χειρισμός diacritics (τόνοι)** είναι κρίσιμος για την εμπειρία χρήστη
- **Character normalization** βελτιώνει σημαντικά την αντιστοίχιση λέξεων
- **Η βαθμολογία με βάση** τη συχνότητα δημιουργεί ελκυστική εξέλιξη δυσκολίας
- **Αποτελεσματικές δομές δεδομένων** (tries, frequency maps) απαραίτητες για απόδοση

Insights δημιουργίας σταυρολέξων:

- Η προσέγγιση ικανοποίησης περιορισμών λειτουργεί καλά για την τοποθέτηση πλέγματος
- Η ευρετική βελτιστοποίηση παρέχει καλή ισορροπία μεταξύ ποιότητας και ταχύτητας
- Απαραίτητες εναλλακτικές στρατηγικές για ακραίες περιπτώσεις και αδύνατες διατάξεις
- Η σταδιακή τοποθέτηση είναι πιο αξιόπιστη από την καθολική βελτιστοποίηση

7.3.2 UI/UX Development Lesson

Ανάπτυξη προσαρμοσμένου Widget:

- **Touch interaction complexity** απαιτεί προσεκτική διαχείριση του state
- **Animation performance** χρειάζεται βελτιστοποίηση για ομαλή εμπειρία
- **Responsive design** πρέπει να δοκιμαστεί σε πολλαπλά μεγέθη οθόνης

State Management Insights:

- **Provider pattern** λειτουργεί καλά για εφαρμογές μεσαίας πολυπλοκότητας
- **Local state vs Global state** η ισορροπία είναι κρίσιμη για την απόδοση
- **Error handling** πρέπει να είναι ολοκληρωμένη και φιλική προς τον χρήστη
- **Real-time updates** απαιτούν προσεκτικές στρατηγικές συγχρονισμού

7.4 Πρακτικές Προκλήσεις που Αντιμετωπίστηκαν

7.4.1 Τεχνικές Προκλήσεις

Επεξεργασία Ελληνικής Γλώσσας:

- **Πρόκληση:** Περιορισμένοι πόροι για την επεξεργασία ελληνικού κειμένου
- **Λύση**: Προσαρμοσμένες υλοποιήσεις για diacritics και ανάλυση συχνότητας
- **Μάθημα**: Συχνά απαιτούνται domain-specific βελτιστοποιήσεις

Βελτιστοποίηση αλγορίθμου:

- **Πρόκληση**: Η πολυπλοκότητα δημιουργίας σταυρολέξων δεν κλιμακώνεται καλά

- **Λύση**: Ευρετικές προσεγγίσεις με λογικούς συμβιβασμούς
- **Μάθημα**: Οι τέλειες λύσεις συχνά δεν είναι πρακτικές - αρκεί το αρκετά καλό

Απόδοση διεπαφής χρήστη:

- **Πρόκληση**: Πολύπλοκα κινούμενα σχέδια που προκαλούν πτώση καρέ
- **Λύση**: Επιλεκτική βελτιστοποίηση και απλοποίηση κινούμενων εικόνων
- **Μάθημα**: Η αντίληψη του χρήστη είναι πιο σημαντική από την τεχνική τελειότητα

7.4.2 Development Process Challenges

Διαχείριση Πεδίου:

- **Πρόκληση**: Feature creep και τάσεις τελειομανίας
- **Λύση**: Focus στο MVP και επαναληπτικές βελτιώσεις
- **Μάθημα**: Ship early, improve continuously

7.5 Ποιοτική Αξιολόγηση της Εφαρμογής

7.5.1 User Experience Assessment

Strengths:

- Διαισθητική διεπαφή που απαιτεί ελάχιστη μάθηση
- Engaging gameplay που ενθαρρύνει τη συνέχιση του παιχνιδιού
- Ομαλή απόδοση σε δοκιμασμένες συσκευές
- Εκπαιδευτική αξία που βελτιώνει το λεξιλόγιο

Areas for Improvement:

- Η εμπειρία ενσωμάτωσης θα μπορούσε να είναι πιο καθοδηγούμενη
- Το σύστημα Hint πρέπει να είναι πιο στρατηγικό
- Η ανατροφοδότηση προόδου θα μπορούσε να είναι πιο παρακινητική
- Η απουσία κοινωνικών χαρακτηριστικών περιορίζει τις δυνατότητες δέσμευσης

7.5.2 Technical Quality Assessment

Code Quality:

- **Clean architecture** με good separation of concerns
- **Maintainable codebase** με clear naming conventions
- **Efficient algorithms** για core functionality
- **Comprehensive error handling** for production stability

Performance:

- **Fast startup time** (< 3 seconds cold start)
- **Responsive interactions** (< 100ms για user actions)

- **Stable memory usage** (< 150MB typical)
- **Reliable network handling** με automatic retries

7.6 Πραγματικές Μελλοντικές Επεκτάσεις

7.6.1 Άμεσες Βελτιώσεις (Επόμενα Βήματα)

Βελτιωμένη Εμπειρία Χρήστη

- **Διαδραστικό εκπαιδευτικό σύστημα**: ενσωμάτωση οδηγού (tutorial) που παρουσιάζει τους βασικούς μηχανισμούς με παραδείγματα και οπτικά βοηθήματα.
- **Σύστημα επιτευγμάτων**: ξεκάθαρα ορόσημα (π.χ. «Πρώτες 10 λύσεις», «Σειρά 7 ημερών») που προσφέρουν σήματα (badges) και εσωτερικά νομίσματα.
- **Ημερήσιες προκλήσεις**: μια νέα πίστα κάθε μέρα με χρονικό περιορισμό, ώστε να αυξάνονται η επαναληψιμότητα και το retention.
- **Έξυπνες υποδείξεις**: δυναμικό σύστημα βοήθειας που προσαρμόζεται στο πλαίσιο της λέξης• προτείνει γράμματα ή παρέχει ετυμολογικούς υπαινιγμούς αντί για απλή αποκάλυψη χαρακτήρων.

Τεχνικές Βελτιώσεις

- **Λειτουργία εκτός σύνδεσης** με τοπική βάση δεδομένων και συγχρονισμό όταν επανέλθει η πρόσβαση στο διαδίκτυο.
- **Βελτιστοποίηση απόδοσης** (CPU / GPU) για παλαιότερες συσκευές· προσαρμογή ποιότητας γραφικών και μείωση all-in-memory δομών.
- **Πληρέστερη ανάλυση χρήσης** με εργαλεία τηλεμετρίας (π.χ. open-source Plausible) ώστε να χαρτογραφούνται «νεκρά» σημεία ροής.
- **Πλαίσιο A/B δοκιμών**: αυτοματοποιημένη εναλλαγή παραλλαγών UI/UX για μέτρηση επιρροής σε KPIs (μονάδα ολοκλήρωσης επιπέδων, χρόνος συνεδρίας κ.λπ.).

7.6.2 Μεσοπρόθεσμες Επεκτάσεις

Επέκταση Περιεχομένου

- **Αυτόματη παραγωγή επιπέδων** μέσω αλγορίθμων και στατιστικών λεξιλογίου, περιορίζοντας τον χειροκίνητο σχεδιασμό.
- **Θεματικές κατηγορίες** (ζώα, γεωγραφία, μυθολογία) που ενισχύουν τον παιγνιώδη χαρακτήρα και τη μαθησιακή αξία.
- **Κλιμακούμενη δυσκολία**: από επίπεδα αρχαρίων μέχρι «Διακεκριμένο Λεξιπλάστη» ώστε να καλύπτονται όλα τα επίπεδα δεξιοτήτων.
- **Προσαρμοσμένες λίστες λέξεων** από τους ίδιους τους χρήστες· δυνατότητα εισαγωγής και κοινής χρήσης custom λεξιθεσιών.

Κοινωνικά Χαρακτηριστικά

- **Πίνακες κατάταξης** (leaderboards) σε τοπικό και παγκόσμιο επίπεδο, με φιλτράρισμα ανά θεματική ενότητα.
- **Μηχανισμός διαμοιρασμού λέξεων** για να προκαλούν οι παίκτες φίλους τους σε ειδικές πίστες.
- **Συνεργατικά ή ανταγωνιστικά multiplayer** (συγχρονισμένα ή ασύγχρονα) για ομαδική επίλυση κρυπτολεξου.
- **Επιλογές σύνδεσης μέσω κοινωνικών δικτύων** (Google, Apple, Facebook) για ταχύτερη εγγραφή και αυτόματη ανάκτηση προόδου.

7.7 Εκπαιδευτική και Προσωπική Αξία

Το έργο λειτουργεί ως εργαστήριο σύνθετης μηχανικής λογισμικού και ταυτόχρονα ως εκπαιδευτικό εργαλείο για τους τελικούς χρήστες. Μέσα από την υλοποίηση:

- Οι χρήστες εμπλουτίζουν λεξιλόγιο, ορθογραφία και ετυμολογικές γνώσεις μέσω παιχνιδιού.
- Ο δημιουργός αξιοποίησε και διεύρυνε τεχνικές και οριζόντιες δεξιότητες, μετατρέποντας τη θεωρία σε απτό προϊόν.

7.7.1 Δεξιότητες που Αναπτύχθηκαν

Τεχνικές Δεξιότητες	Οριζόντιες (Soft) Δεξιότητες
• Προχωρημένη ανάπτυξη Flutter και δημιουργία custom widgets	• Διαχείριση έργου (agile σπριντ, road-mapping)
• Διασύνδεση backend (Supabase, PostgreSQL)	• Επίλυση προβλημάτων σε μη ορισμένα πλαίσια
• Βελτιστοποίηση αλγορίθμων (τοποθέτηση λέξεων, αναζήτηση)	• Ερευνητικές ικανότητες για αξιολόγηση τεχνολογιών
• Σχεδίαση και ασφάλεια βάσεων δεδομένων	• Τεκμηρίωση & τεχνική συγγραφή (wiki, blog)
• Σχεδιασμός UI/UX με προσβασιμότητα (WCAG)	• Διαχείριση χρόνου και τήρηση προθεσμιών

7.8 Συνεισφορά στην Τοπική Κοινότητα Προγραμματιστών

Η διάθεση του κώδικα, των άρθρων και των παρουσιάσεων ενισχύει εμπρακτα την ελληνική κοινότητα λογισμικού. Μέσα από συναντήσεις (meetups), διαδικτυακά σεμινάρια και ανοιχτά repos, το έργο:

- Παρέχει παραδείγματα αρχιτεκτονικής Flutter με Supabase, χρήσιμα για εκπαιδευτικούς και επαγγελματίες.
- Προωθεί τη συνεργατική μάθηση, όπου νέοι προγραμματιστές συνεισφέρουν μεταφράσεις, pull requests και βελτιώσεις.
- Δικτυώνει δημιουργούς, καθηγητές και μαθητές σε κοινό σκοπό: τη διάσωση και ανάδειξη της ελληνικής γλώσσας μέσω τεχνολογίας.

7.8.1 Open Source Potential

Επαναχρησιμοποιούμενα Συστατικά	Πιθανές Χρήσεις
Letter Connector widget	Οποιοδήποτε παιχνίδι τύπου word-search ή crossword
Βοηθητικά εργαλεία επεξεργασίας ελληνικού κειμένου	Εφαρμογές ορθογραφικού ελέγχου, εκπαιδευτικά apps
Αλγόριθμοι παραγωγής σταυρολέξων	Προσαρμογή σε άλλες γλώσσες ή θεματικά παιχνίδια
Πρότυπα διασύνδεσης Flutter-Supabase	Ταχεία εκκίνηση (boilerplate) για νεοσύστατες εφαρμογές

Θέτοντας τα παραπάνω σε δημόσια αποθετήρια (GitHub), η κοινότητα:

- Κερδίζει εργαλεία κορυφαίας ποιότητας για δικές της ανάγκες.
- Εξελίσσει συλλογικά το έργο, διορθώνοντας σφάλματα και προτείνοντας βελτιώσεις.
- Διαδίδει καλές πρακτικές ανάπτυξης, από δοκιμές μονάδας έως αυτοματοποιημένο CI.

7.10 Τελικές Σκέψεις

7.10.1 Long-term Vision

Sustainable Development: Η εφαρμογή στηρίζεται σε στιβαρή, **αρθρωτή αρχιτεκτονική** που επιτρέπει **σταδιακές βελτιώσεις** χωρίς τη χρονοβόρα και ριψοκίνδυνη **αναδόμηση κώδικα**. Με διακριτά επίπεδα (δεδομένων, λογικής, παρουσίασης) και καλά ορισμένες διεπαφές, μπορεί να εξελίσσεται οργανικά – από μικρές διορθώσεις μέχρι μεγάλες νέες δυνατότητες – με ελάχιστο τεχνικό χρέος.

Community Building: Με τη σωστή υποστήριξη, το έργο μπορεί να μετατραπεί σε κόμβο για τους λάτρεις της ελληνικής γλώσσας, δημιουργώντας περιεχόμενο, ανταλλάσσοντας ιδέες και συνεισφέροντας κώδικα. Έτσι ενισχύεται η διατήρηση και προώθηση της ελληνικής γλώσσας στην ψηφιακή εποχή και καλλιεργείται μια κοινότητα που θα τροφοδοτεί την εφαρμογή με νέο υλικό και καινοτομίες.

Personal Growth: Το έργο λειτουργεί ως ορόσημο που αποδεικνύει ικανότητες σε σύνθετη ανάπτυξη λογισμικού και προετοιμάζει για απαιτητικούς επαγγελματικούς ρόλους μηχανικού λογισμικού. Η κατανόηση της πλήρους αλυσίδας – από τον σχεδιασμό αλγορίθμων έως τη διανομή σε καταστήματα εφαρμογών – αποτελεί ακλόνητο θεμέλιο για μελλοντικές, ακόμη πιο φιλόδοξες προκλήσεις.

7.11 Συμπέρασμα

Η ανάπτυξη του κλώνου Words of Wonders ολοκληρώθηκε με επιτυχία, επιτυγχάνοντας όλους τους αρχικούς στόχους και θέτοντας μια δυνατή βάση για περαιτέρω εξέλιξη. Ο συνδυασμός τεχνικής

αρτιότητας, σχεδίασης με επίκεντρο τον χρήστη και πολιτισμικής συνεισφοράς καθιστά το έργο πολύτιμη προσθήκη τόσο στο προσωπικό portfolio όσο και στη διεθνή – πλέον – ελληνική κοινότητα προγραμματιστών.

Κύρια Επιτεύγματα

- Πλήρως λειτουργική διαπλατφορμική εφαρμογή, με ποιότητα έτοιμη για παραγωγή.
- Καινοτόμοι αλγόριθμοι για επεξεργασία της ελληνικής γλώσσας (ανάλυση λέξεων, τοποθέτηση σε πλέγμα, επαλήθευση λύσεων).
- Υιοθέτηση σύγχρονων πρακτικών ανάπτυξης με εργαλεία προδιαγραφών βιομηχανίας (CI/CD, δοκιμές μονάδας, συνεχής έλεγχος ποιότητας κώδικα).
- Εκπαιδευτική αξία τόσο για τους τελικούς χρήστες όσο και για άλλους προγραμματιστές που μελετούν το αποθετήριο.
- Πολιτισμική συνεισφορά στη διάσωση και ανάδειξη της ελληνικής γλώσσας μέσω ψηφιακού παιχνιδιού.

Προοπτικές Εξέλιξης

Το έργο αποτελεί εφαλτήριο για:

1. **Νέα χαρακτηριστικά** (π.χ. multiplayer, καθημερινές προκλήσεις) βασισμένα στο υπάρχον αρθρωτό υπόβαθρο.

2. **Ερευνητικές εφαρμογές** στην επεξεργασία φυσικής γλώσσας (NLP) με εστίαση στα ελληνικά.

3. **Επαγγελματική δικτύωση** και συνεργασίες με εκπαιδευτικούς φορείς, εκδότες λεξικών ή εταιρείες ed-tech.

Η εμπειρία, οι δεξιότητες και, κυρίως, η αυτοπεποίθηση που αποκομίστηκαν αποτελούν **ανεκτίμητο εφόδιο** για τη μελλοντική επαγγελματική πορεία. Με αφετηρία αυτό το έργο, ανοίγει ο δρόμος για ακόμη **μεγαλύτερα, καινοτόμα εγχειρήματα** που θα συνδυάζουν τεχνολογία, εκπαίδευση και πολιτισμό.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] "Why you should Flutter over React Native - A comprehensive perspective" https://dev.to/pranta/why-you-should-flutter-over-react-native-a-comprehensive-perspective-1bo2

[2] "React Native vs Flutter: What to Choose in 2024?", softteco.com, https://softteco.com/blog/react-native-vs-flutter

[4] Pub.dev. "Package statistics – total packages." Accessed 5 May 2025. https://pub.dev

[5] Google. "Flutter Performance: Bridgeless Rendering vs React Native Bridge." *Flutter Docs*, rev. 2024-12.

[6] Google. "Hot Reload." *Flutter Documentation v3.22*, Accessed 5 May 2025.

[7] Ίδρυμα Μανόλη Τριανταφυλλίδη, "Λεξικό της Κοινής Νεοελληνικής", 2024. [Online]. Available: http://www.greek-language.gr/greekLang/modern_greek/tools/lexica/triantafyllides/

[8] Κέντρο Ελληνικής Γλώσσας, "Λεξικό της Κοινής Νεοελληνικής (ΛΚΝ)", 2024.

[9] OpenThesaurus.gr, "Ελληνική βάση συνωνύμων", 2024. [Online]. Available: https://www.openthesaurus.gr/

[10] D. E. Goldberg, "Genetic Algorithms in Search, Optimization, and Machine Learning," Addison-Wesley, 1989.

[11] S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi, "Optimization by Simulated Annealing," Science, vol. 220, no. 4598, pp. 671-680, May 1983.

[12] https://en.wikipedia.org/wiki/Spoke%E2%80%93hub_distribution_paradigm