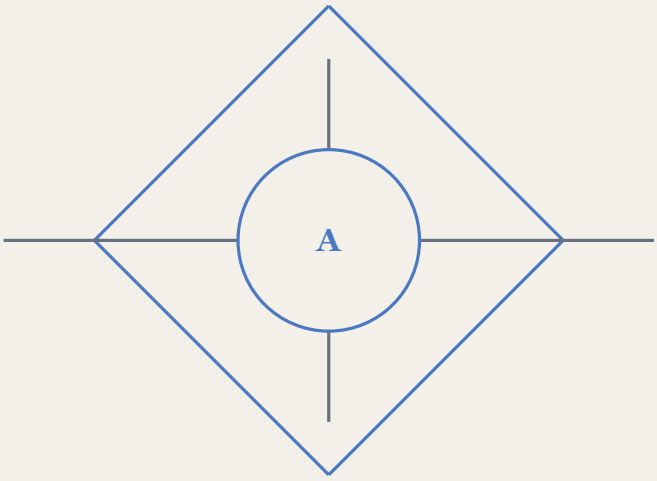
GNOSERA

# App Engineering με AI

Από την ιδέα στο App Store σε 30 ημέρες




KNOWLEDGE FOR

THE NEXT ERA

by **INDIGO**

ACADEMIC EDITION

FULL RESEARCH EDITION · 2026

**15 ΚΕΦΑΛΑΙΑ · 75 ΕΝΟΤΗΤΕΣ**

# Πώς να χρησιμοποιήσεις αυτό το βιβλίο

Η έκδοση έχει σχεδιαστεί ως εφαρμοσμένο πρόγραμμα μάθησης. Στόχος δεν είναι η παθητική ανάγνωση, αλλά η μετατροπή κάθε κεφαλαίου σε συγκεκριμένη απόφαση, έλεγχο ή παραδοτέο.

**01** **Κατανόησε**

Διάβασε πρώτα τον μηχανισμό, τους βασικούς όρους και τα όρια της μεθόδου.

**02** **Εφάρμοσε**

Χρησιμοποίησε δικά σου δεδομένα ή ένα ασφαλές υποθετικό σενάριο στο εργαστήριο.

**03** **Έλεγξε**

Δοκίμασε κανονικές και δυσμενείς συνθήκες. Κατέγραψε παραδοχές και πηγές.

**04** **Αποφάσισε**

Όρισε ενέργεια, υπεύθυνο, προθεσμία και ημερομηνία επανεξέτασης.

# Πώς χρησιμοποιείται αυτό το βιβλίο

Το βιβλίο είναι σχεδιασμένο ως ταχύρυθμο πρόγραμμα εφαρμοσμένης μηχανικής λογισμικού. Δεν χρειάζεται να απομνημονεύσεις όλο τον κώδικα. Χρειάζεται να κατανοείς τις αποφάσεις, να ελέγχεις το αποτέλεσμα και να μπορείς να συνεργάζεσαι αποτελεσματικά με το Codex ή έναν developer.

| | |
|---|---|
| **Πρόγραμμα 30 ημερών** | Μελέτησε μισό κεφάλαιο κάθε ημέρα. Κάνε το εργαστήριο πριν κοιτάξεις τη λύση. Κράτα ένα πραγματικό project ανοικτό σε όλη τη διάρκεια. |

## Η μέθοδος

| | | |
|---|---|---|
| Κατανόηση | Διαβάζεις το μοντέλο και τους κανόνες. | Μπορείς να το εξηγήσεις απλά. |
| Εφαρμογή | Χτίζεις το μικρό παράδειγμα. | Λειτουργικό αποτέλεσμα. |
| Έλεγχος | Δοκιμάζεις happy path και αποτυχίες. | Αποδείξεις, όχι υποθέσεις. |
| Ανασκόπηση | Χρησιμοποιείς checklist και quiz. | Καθαρά κενά για επανάληψη. |

# Περιεχόμενα

ΚΕΦΑΛΑΙΟ 01

# Πώς λειτουργεί ένα app

Να βλέπεις την εφαρμογή σαν ενιαίο σύστημα και να εντοπίζεις τις ευθύνες κάθε μέρους.

| | |
|---|---|
| **Στο τέλος** | Θα μπορείς να εφαρμόσεις το κεφάλαιο σε πραγματικό project, να εξηγήσεις τις επιλογές σου και να αναγνωρίσεις τα βασικά σημεία κινδύνου. |

## Διαδρομή κεφαλαίου

# 1.1 Frontend και εμπειρία χρήστη

Η ενότητα Frontend και εμπειρία χρήστη αποτελεί βασικό δομικό στοιχείο του κεφαλαίου «Πώς λειτουργεί ένα app». Στόχος δεν είναι η μηχανική απομνημόνευση, αλλά να μπορείς να πάρεις σωστή απόφαση όταν το ίδιο ζήτημα εμφανιστεί μέσα σε πραγματική εφαρμογή.

## Το νοητικό μοντέλο

Ξεκίνα ορίζοντας τι εισέρχεται στη λειτουργία, ποιος έχει δικαίωμα να την ενεργοποιήσει, ποιος κανόνας εφαρμόζεται και ποιο αποτέλεσμα πρέπει να παραχθεί. Στο θέμα «Frontend και εμπειρία χρήστη» η καθαρή ευθύνη είναι σημαντικότερη από την ποσότητα κώδικα. Αν μία λειτουργία κάνει πολλά άσχετα πράγματα, διαίρεσέ την σε μικρότερα βήματα με σαφείς εισόδους και εξόδους.

## Πρακτικοί κανόνες

- Περιέγραψε πρώτα το αναμενόμενο αποτέλεσμα για το «Frontend και εμπειρία χρήστη».
- Ξεχώρισε δεδομένα, κανόνες, παρουσίαση και εξωτερικές εξαρτήσεις.
- Έλεγξε κενές τιμές, καθυστέρηση, διπλή ενέργεια και αποτυχία δικτύου.
- Μην εμπιστεύεσαι δεδομένα χρήστη πριν από validation και authorization.

## Μικρό παράδειγμα

Σε ένα e-shop, εφάρμοσε το «Frontend και εμπειρία χρήστη» στη ροή αγοράς: ο χρήστης επιλέγει προϊόν, το σύστημα επαληθεύει τα δεδομένα, εκτελεί τη λειτουργία μία φορά και επιστρέφει σαφή κατάσταση επιτυχίας ή αποτυχίας. Το αποτέλεσμα πρέπει να παραμένει προβλέψιμο ακόμη και αν ο χρήστης πατήσει δύο φορές ή η σύνδεση διακοπεί.

| **Έλεγχος κατανόησης** | Μπορείς να εξηγήσεις το «Frontend και εμπειρία χρήστη» χωρίς ορολογία, να δώσεις ένα παράδειγμα και να ονομάσεις δύο τρόπους αποτυχίας; |
|---|---|

## Prompt εφαρμογής

```
Εξέτασε το project μου ως προς: Frontend και εμπειρία χρήστη.
Δώσε: τρέχουσα ροή, κινδύνους, προτεινόμενη αλλαγή,
κριτήρια αποδοχής και ελάχιστες δοκιμές.
Μην αλλάξεις κώδικα πριν παρουσιάσεις το σχέδιο.
```

# 1.2 Backend και επιχειρησιακοί κανόνες

Η ενότητα Backend και επιχειρησιακοί κανόνες αποτελεί βασικό δομικό στοιχείο του κεφαλαίου «Πώς λειτουργεί ένα app». Στόχος δεν είναι η μηχανική απομνημόνευση, αλλά να μπορείς να πάρεις σωστή απόφαση όταν το ίδιο ζήτημα εμφανιστεί μέσα σε πραγματική εφαρμογή.

## Το νοητικό μοντέλο

Ξεκίνα ορίζοντας τι εισέρχεται στη λειτουργία, ποιος έχει δικαίωμα να την ενεργοποιήσει, ποιος κανόνας εφαρμόζεται και ποιο αποτέλεσμα πρέπει να παραχθεί. Στο θέμα «Backend και επιχειρησιακοί κανόνες» η καθαρή ευθύνη είναι σημαντικότερη από την ποσότητα κώδικα. Αν μία λειτουργία κάνει πολλά άσχετα πράγματα, διαίρεσέ την σε μικρότερα βήματα με σαφείς εισόδους και εξόδους.

## Πρακτικοί κανόνες

- Περιέγραψε πρώτα το αναμενόμενο αποτέλεσμα για το «Backend και επιχειρησιακοί κανόνες».
- Ξεχώρισε δεδομένα, κανόνες, παρουσίαση και εξωτερικές εξαρτήσεις.
- Έλεγξε κενές τιμές, καθυστέρηση, διπλή ενέργεια και αποτυχία δικτύου.
- Μην εμπιστεύεσαι δεδομένα χρήστη πριν από validation και authorization.

## Μικρό παράδειγμα

Σε ένα e-shop, εφάρμοσε το «Backend και επιχειρησιακοί κανόνες» στη ροή αγοράς: ο χρήστης επιλέγει προϊόν, το σύστημα επαληθεύει τα δεδομένα, εκτελεί τη λειτουργία μία φορά και επιστρέφει σαφή κατάσταση επιτυχίας ή αποτυχίας. Το αποτέλεσμα πρέπει να παραμένει προβλέψιμο ακόμη και αν ο χρήστης πατήσει δύο φορές ή η σύνδεση διακοπεί.

| **Έλεγχος κατανόησης** | Μπορείς να εξηγήσεις το «Backend και επιχειρησιακοί κανόνες» χωρίς ορολογία, να δώσεις ένα παράδειγμα και να ονομάσεις δύο τρόπους αποτυχίας; |
|---|---|

## Prompt εφαρμογής

```
Εξέτασε το project μου ως προς: Backend και επιχειρησιακοί κανόνες.
Δώσε: τρέχουσα ροή, κινδύνους, προτεινόμενη αλλαγή,
κριτήρια αποδοχής και ελάχιστες δοκιμές.
Μην αλλάξεις κώδικα πριν παρουσιάσεις το σχέδιο.
```

# 1.3 Database και σχέσεις δεδομένων

Η ενότητα Database και σχέσεις δεδομένων αποτελεί βασικό δομικό στοιχείο του κεφαλαίου «Πώς λειτουργεί ένα app». Στόχος δεν είναι η μηχανική απομνημόνευση, αλλά να μπορείς να πάρεις σωστή απόφαση όταν το ίδιο ζήτημα εμφανιστεί μέσα σε πραγματική εφαρμογή.

## Το νοητικό μοντέλο

Ξεκίνα ορίζοντας τι εισέρχεται στη λειτουργία, ποιος έχει δικαίωμα να την ενεργοποιήσει, ποιος κανόνας εφαρμόζεται και ποιο αποτέλεσμα πρέπει να παραχθεί. Στο θέμα «Database και σχέσεις δεδομένων» η καθαρή ευθύνη είναι σημαντικότερη από την ποσότητα κώδικα. Αν μία λειτουργία κάνει πολλά άσχετα πράγματα, διαίρεσέ την σε μικρότερα βήματα με σαφείς εισόδους και εξόδους.

## Πρακτικοί κανόνες

- Περιέγραψε πρώτα το αναμενόμενο αποτέλεσμα για το «Database και σχέσεις δεδομένων».
- Ξεχώρισε δεδομένα, κανόνες, παρουσίαση και εξωτερικές εξαρτήσεις.
- Έλεγξε κενές τιμές, καθυστέρηση, διπλή ενέργεια και αποτυχία δικτύου.
- Μην εμπιστεύεσαι δεδομένα χρήστη πριν από validation και authorization.

## Μικρό παράδειγμα

Σε ένα e-shop, εφάρμοσε το «Database και σχέσεις δεδομένων» στη ροή αγοράς: ο χρήστης επιλέγει προϊόν, το σύστημα επαληθεύει τα δεδομένα, εκτελεί τη λειτουργία μία φορά και επιστρέφει σαφή κατάσταση επιτυχίας ή αποτυχίας. Το αποτέλεσμα πρέπει να παραμένει προβλέψιμο ακόμη και αν ο χρήστης πατήσει δύο φορές ή η σύνδεση διακοπεί.

| | |
|---|---|
| **Έλεγχος κατανόησης** | Μπορείς να εξηγήσεις το «Database και σχέσεις δεδομένων» χωρίς ορολογία, να δώσεις ένα παράδειγμα και να ονομάσεις δύο τρόπους αποτυχίας; |

## Prompt εφαρμογής

```
Εξέτασε το project μου ως προς: Database και σχέσεις δεδομένων.
Δώσε: τρέχουσα ροή, κινδύνους, προτεινόμενη αλλαγή,
κριτήρια αποδοχής και ελάχιστες δοκιμές.
Μην αλλάξεις κώδικα πριν παρουσιάσεις το σχέδιο.
```

# 1.4 APIs, requests και responses

Η ενότητα APIs, requests και responses αποτελεί βασικό δομικό στοιχείο του κεφαλαίου «Πώς λειτουργεί ένα app». Στόχος δεν είναι η μηχανική απομνημόνευση, αλλά να μπορείς να πάρεις σωστή απόφαση όταν το ίδιο ζήτημα εμφανιστεί μέσα σε πραγματική εφαρμογή.

## Το νοητικό μοντέλο

Ξεκίνα ορίζοντας τι εισέρχεται στη λειτουργία, ποιος έχει δικαίωμα να την ενεργοποιήσει, ποιος κανόνας εφαρμόζεται και ποιο αποτέλεσμα πρέπει να παραχθεί. Στο θέμα «APIs, requests και responses» η καθαρή ευθύνη είναι σημαντικότερη από την ποσότητα κώδικα. Αν μία λειτουργία κάνει πολλά άσχετα πράγματα, διαίρεσέ την σε μικρότερα βήματα με σαφείς εισόδους και εξόδους.

## Πρακτικοί κανόνες

- Περιέγραψε πρώτα το αναμενόμενο αποτέλεσμα για το «APIs, requests και responses».
- Ξεχώρισε δεδομένα, κανόνες, παρουσίαση και εξωτερικές εξαρτήσεις.
- Έλεγξε κενές τιμές, καθυστέρηση, διπλή ενέργεια και αποτυχία δικτύου.
- Μην εμπιστεύεσαι δεδομένα χρήστη πριν από validation και authorization.

## Μικρό παράδειγμα

Σε ένα e-shop, εφάρμοσε το «APIs, requests και responses» στη ροή αγοράς: ο χρήστης επιλέγει προϊόν, το σύστημα επαληθεύει τα δεδομένα, εκτελεί τη λειτουργία μία φορά και επιστρέφει σαφή κατάσταση επιτυχίας ή αποτυχίας. Το αποτέλεσμα πρέπει να παραμένει προβλέψιμο ακόμη και αν ο χρήστης πατήσει δύο φορές ή η σύνδεση διακοπεί.

| **Έλεγχος κατανόησης** | Μπορείς να εξηγήσεις το «APIs, requests και responses» χωρίς ορολογία, να δώσεις ένα παράδειγμα και να ονομάσεις δύο τρόπους αποτυχίας; |
|---|---|

## Prompt εφαρμογής

```
Εξέτασε το project μου ως προς: APIs, requests και responses.
Δώσε: τρέχουσα ροή, κινδύνους, προτεινόμενη αλλαγή,
κριτήρια αποδοχής και ελάχιστες δοκιμές.
Μην αλλάξεις κώδικα πριν παρουσιάσεις το σχέδιο.
```

# 1.5 State, authentication και όρια εμπιστοσύνης

Η ενότητα State, authentication και όρια εμπιστοσύνης αποτελεί βασικό δομικό στοιχείο του κεφαλαίου «Πώς λειτουργεί ένα app». Στόχος δεν είναι η μηχανική απομνημόνευση, αλλά να μπορείς να πάρεις σωστή απόφαση όταν το ίδιο ζήτημα εμφανιστεί μέσα σε πραγματική εφαρμογή.

## Το νοητικό μοντέλο

Ξεκίνα ορίζοντας τι εισέρχεται στη λειτουργία, ποιος έχει δικαίωμα να την ενεργοποιήσει, ποιος κανόνας εφαρμόζεται και ποιο αποτέλεσμα πρέπει να παραχθεί. Στο θέμα «State, authentication και όρια εμπιστοσύνης» η καθαρή ευθύνη είναι σημαντικότερη από την ποσότητα κώδικα. Αν μία λειτουργία κάνει πολλά άσχετα πράγματα, διαίρεσέ την σε μικρότερα βήματα με σαφείς εισόδους και εξόδους.

## Πρακτικοί κανόνες

- Περιέγραψε πρώτα το αναμενόμενο αποτέλεσμα για το «State, authentication και όρια εμπιστοσύνης».
- Ξεχώρισε δεδομένα, κανόνες, παρουσίαση και εξωτερικές εξαρτήσεις.
- Έλεγξε κενές τιμές, καθυστέρηση, διπλή ενέργεια και αποτυχία δικτύου.
- Μην εμπιστεύεσαι δεδομένα χρήστη πριν από validation και authorization.

## Μικρό παράδειγμα

Σε ένα e-shop, εφάρμοσε το «State, authentication και όρια εμπιστοσύνης» στη ροή αγοράς: ο χρήστης επιλέγει προϊόν, το σύστημα επαληθεύει τα δεδομένα, εκτελεί τη λειτουργία μία φορά και επιστρέφει σαφή κατάσταση επιτυχίας ή αποτυχίας. Το αποτέλεσμα πρέπει να παραμένει προβλέψιμο ακόμη και αν ο χρήστης πατήσει δύο φορές ή η σύνδεση διακοπεί.

| | |
|---|---|
| **Έλεγχος κατανόησης** | Μπορείς να εξηγήσεις το «State, authentication και όρια εμπιστοσύνης» χωρίς ορολογία, να δώσεις ένα παράδειγμα και να ονομάσεις δύο τρόπους αποτυχίας; |

## Prompt εφαρμογής

```
Εξέτασε το project μου ως προς: State, authentication και όρια εμπιστοσύνης.
Δώσε: τρέχουσα ροή, κινδύνους, προτεινόμενη αλλαγή,
κριτήρια αποδοχής και ελάχιστες δοκιμές.
Μην αλλάξεις κώδικα πριν παρουσιάσεις το σχέδιο.
```

# 1.6 Εργαστήριο κεφαλαίου

Σχεδίασε τη ροή μιας παραγγελίας από το tap μέχρι την αποθήκευση.

## Οδηγίες

- Γράψε το happy path σε 5–8 αριθμημένα βήματα.
- Κατάγραψε τουλάχιστον τέσσερις αποτυχίες ή ακραίες περιπτώσεις.
- Όρισε source of truth και ποιο μέρος έχει κάθε ευθύνη.
- Ζήτησε από το Codex σχέδιο και έλεγξέ το πριν επιτρέψεις αλλαγές.
- Υλοποίησε τη μικρότερη λειτουργική έκδοση και δοκίμασέ την.

## Παράδειγμα αναφοράς

```
GET /api/products
→ backend validation
→ database query
→ JSON response
```

## Definition of Done

| | |
|---|---|
| Λειτουργία | Το βασικό σενάριο περνά από άκρη σε άκρη. |
| Αποτυχίες | Ο χρήστης λαμβάνει χρήσιμα μηνύματα χωρίς απώλεια δεδομένων. |
| Ασφάλεια | Secrets και αποφάσεις δικαιωμάτων μένουν στον server. |
| Δοκιμές | Υπάρχει επαναλήψιμος έλεγχος για τα κρίσιμα σημεία. |
| Τεκμηρίωση | Άλλος άνθρωπος μπορεί να καταλάβει τη ροή. |

# 1.7 Quiz και checklist

Απάντησε πρώτα χωρίς να κοιτάξεις πίσω.

**Τι πρόβλημα λύνει το «Frontend και εμπειρία χρήστη»;**

.....

**Ποια δεδομένα θεωρούνται αξιόπιστα στο «Πώς λειτουργεί ένα app»;**

.....

**Ποια ευθύνη ανήκει στον client και ποια στον server;**

.....

**Ποιες δύο αποτυχίες πρέπει οπωσδήποτε να δοκιμαστούν;**

.....

**Πώς αποδεικνύεις ότι η υλοποίηση λειτουργεί;**

.....

## Checklist

- Μπορώ να εξηγήσω τα πέντε βασικά θέματα του κεφαλαίου 1.
- Έκανα το εργαστήριο σε πραγματικό ή δοκιμαστικό project.
- Δοκίμασα τουλάχιστον μία αποτυχία, όχι μόνο το happy path.
- Έγραψα καθαρά κριτήρια αποδοχής.
- Ξέρω ποιο είναι το επόμενο μικρό βήμα.

| | |
|---|---|
| **Κανόνας προόδου** | Αν μπορείς να εξηγήσεις, να εφαρμόσεις και να ελέγξεις, προχώρα. Αν απλώς αναγνωρίζεις τους όρους, επανάλαβε το εργαστήριο. |

ΚΕΦΑΛΑΙΟ 02

# Βασικές αρχές κώδικα

Να διαβάζεις μικρά προγράμματα και να μετατρέπεις επιχειρησιακούς κανόνες σε σαφή λογική.

| | |
|---|---|
| **Στο τέλος** | Θα μπορείς να εφαρμόσεις το κεφάλαιο σε πραγματικό project, να εξηγήσεις τις επιλογές σου και να αναγνωρίσεις τα βασικά σημεία κινδύνου. |

## Διαδρομή κεφαλαίου

- 1. Μεταβλητές και τύποι τιμών
- 2. Συνθήκες και αποφάσεις
- 3. Loops και επανάληψη
- 4. Συναρτήσεις και παράμετροι
- 5. Arrays, objects και async ροές

# 2.1 Μεταβλητές και τύποι τιμών

Η ενότητα Μεταβλητές και τύποι τιμών αποτελεί βασικό δομικό στοιχείο του κεφαλαίου «Βασικές αρχές κώδικα». Στόχος δεν είναι η μηχανική απομνημόνευση, αλλά να μπορείς να πάρεις σωστή απόφαση όταν το ίδιο ζήτημα εμφανιστεί μέσα σε πραγματική εφαρμογή.

## Το νοητικό μοντέλο

Ξεκίνα ορίζοντας τι εισέρχεται στη λειτουργία, ποιος έχει δικαίωμα να την ενεργοποιήσει, ποιος κανόνας εφαρμόζεται και ποιο αποτέλεσμα πρέπει να παραχθεί. Στο θέμα «Μεταβλητές και τύποι τιμών» η καθαρή ευθύνη είναι σημαντικότερη από την ποσότητα κώδικα. Αν μία λειτουργία κάνει πολλά άσχετα πράγματα, διαίρεσέ την σε μικρότερα βήματα με σαφείς εισόδους και εξόδους.

## Πρακτικοί κανόνες

- Περιέγραψε πρώτα το αναμενόμενο αποτέλεσμα για το «Μεταβλητές και τύποι τιμών».
- Ξεχώρισε δεδομένα, κανόνες, παρουσίαση και εξωτερικές εξαρτήσεις.
- Έλεγξε κενές τιμές, καθυστέρηση, διπλή ενέργεια και αποτυχία δικτύου.
- Μην εμπιστεύεσαι δεδομένα χρήστη πριν από validation και authorization.

## Μικρό παράδειγμα

Σε ένα e-shop, εφάρμοσε το «Μεταβλητές και τύποι τιμών» στη ροή αγοράς: ο χρήστης επιλέγει προϊόν, το σύστημα επαληθεύει τα δεδομένα, εκτελεί τη λειτουργία μία φορά και επιστρέφει σαφή κατάσταση επιτυχίας ή αποτυχίας. Το αποτέλεσμα πρέπει να παραμένει προβλέψιμο ακόμη και αν ο χρήστης πατήσει δύο φορές ή η σύνδεση διακοπεί.

| **Έλεγχος κατανόησης** | Μπορείς να εξηγήσεις το «Μεταβλητές και τύποι τιμών» χωρίς ορολογία, να δώσεις ένα παράδειγμα και να ονομάσεις δύο τρόπους αποτυχίας; |
|---|---|

## Prompt εφαρμογής

```
Εξέτασε το project μου ως προς: Μεταβλητές και τύποι τιμών.
Δώσε: τρέχουσα ροή, κινδύνους, προτεινόμενη αλλαγή,
κριτήρια αποδοχής και ελάχιστες δοκιμές.
Μην αλλάξεις κώδικα πριν παρουσιάσεις το σχέδιο.
```

# 2.2 Συνθήκες και αποφάσεις

Η ενότητα Συνθήκες και αποφάσεις αποτελεί βασικό δομικό στοιχείο του κεφαλαίου «Βασικές αρχές κώδικα». Στόχος δεν είναι η μηχανική απομνημόνευση, αλλά να μπορείς να πάρεις σωστή απόφαση όταν το ίδιο ζήτημα εμφανιστεί μέσα σε πραγματική εφαρμογή.

## Το νοητικό μοντέλο

Ξεκίνα ορίζοντας τι εισέρχεται στη λειτουργία, ποιος έχει δικαίωμα να την ενεργοποιήσει, ποιος κανόνας εφαρμόζεται και ποιο αποτέλεσμα πρέπει να παραχθεί. Στο θέμα «Συνθήκες και αποφάσεις» η καθαρή ευθύνη είναι σημαντικότερη από την ποσότητα κώδικα. Αν μία λειτουργία κάνει πολλά άσχετα πράγματα, διαίρεσέ την σε μικρότερα βήματα με σαφείς εισόδους και εξόδους.

## Πρακτικοί κανόνες

- Περιέγραψε πρώτα το αναμενόμενο αποτέλεσμα για το «Συνθήκες και αποφάσεις».
- Ξεχώρισε δεδομένα, κανόνες, παρουσίαση και εξωτερικές εξαρτήσεις.
- Έλεγξε κενές τιμές, καθυστέρηση, διπλή ενέργεια και αποτυχία δικτύου.
- Μην εμπιστεύεσαι δεδομένα χρήστη πριν από validation και authorization.

## Μικρό παράδειγμα

Σε ένα e-shop, εφάρμοσε το «Συνθήκες και αποφάσεις» στη ροή αγοράς: ο χρήστης επιλέγει προϊόν, το σύστημα επαληθεύει τα δεδομένα, εκτελεί τη λειτουργία μία φορά και επιστρέφει σαφή κατάσταση επιτυχίας ή αποτυχίας. Το αποτέλεσμα πρέπει να παραμένει προβλέψιμο ακόμη και αν ο χρήστης πατήσει δύο φορές ή η σύνδεση διακοπεί.

| | |
|---|---|
| **Έλεγχος κατανόησης** | Μπορείς να εξηγήσεις το «Συνθήκες και αποφάσεις» χωρίς ορολογία, να δώσεις ένα παράδειγμα και να ονομάσεις δύο τρόπους αποτυχίας; |

## Prompt εφαρμογής

```
Εξέτασε το project μου ως προς: Συνθήκες και αποφάσεις.
Δώσε: τρέχουσα ροή, κινδύνους, προτεινόμενη αλλαγή,
κριτήρια αποδοχής και ελάχιστες δοκιμές.
Μην αλλάξεις κώδικα πριν παρουσιάσεις το σχέδιο.
```

# 2.3 Loops και επανάληψη

Η ενότητα Loops και επανάληψη αποτελεί βασικό δομικό στοιχείο του κεφαλαίου «Βασικές αρχές κώδικα». Στόχος δεν είναι η μηχανική απομνημόνευση, αλλά να μπορείς να πάρεις σωστή απόφαση όταν το ίδιο ζήτημα εμφανιστεί μέσα σε πραγματική εφαρμογή.

## Το νοητικό μοντέλο

Ξεκίνα ορίζοντας τι εισέρχεται στη λειτουργία, ποιος έχει δικαίωμα να την ενεργοποιήσει, ποιος κανόνας εφαρμόζεται και ποιο αποτέλεσμα πρέπει να παραχθεί. Στο θέμα «Loops και επανάληψη» η καθαρή ευθύνη είναι σημαντικότερη από την ποσότητα κώδικα. Αν μία λειτουργία κάνει πολλά άσχετα πράγματα, διαίρεσέ την σε μικρότερα βήματα με σαφείς εισόδους και εξόδους.

## Πρακτικοί κανόνες

- Περιέγραψε πρώτα το αναμενόμενο αποτέλεσμα για το «Loops και επανάληψη».
- Ξεχώρισε δεδομένα, κανόνες, παρουσίαση και εξωτερικές εξαρτήσεις.
- Έλεγξε κενές τιμές, καθυστέρηση, διπλή ενέργεια και αποτυχία δικτύου.
- Μην εμπιστεύεσαι δεδομένα χρήστη πριν από validation και authorization.

## Μικρό παράδειγμα

Σε ένα e-shop, εφάρμοσε το «Loops και επανάληψη» στη ροή αγοράς: ο χρήστης επιλέγει προϊόν, το σύστημα επαληθεύει τα δεδομένα, εκτελεί τη λειτουργία μία φορά και επιστρέφει σαφή κατάσταση επιτυχίας ή αποτυχίας. Το αποτέλεσμα πρέπει να παραμένει προβλέψιμο ακόμη και αν ο χρήστης πατήσει δύο φορές ή η σύνδεση διακοπεί.

| **Έλεγχος κατανόησης** | Μπορείς να εξηγήσεις το «Loops και επανάληψη» χωρίς ορολογία, να δώσεις ένα παράδειγμα και να ονομάσεις δύο τρόπους αποτυχίας; |
|---|---|

## Prompt εφαρμογής

```
Εξέτασε το project μου ως προς: Loops και επανάληψη.
Δώσε: τρέχουσα ροή, κινδύνους, προτεινόμενη αλλαγή,
κριτήρια αποδοχής και ελάχιστες δοκιμές.
Μην αλλάξεις κώδικα πριν παρουσιάσεις το σχέδιο.
```

# 2.4 Συναρτήσεις και παράμετροι

Η ενότητα Συναρτήσεις και παράμετροι αποτελεί βασικό δομικό στοιχείο του κεφαλαίου «Βασικές αρχές κώδικα». Στόχος δεν είναι η μηχανική απομνημόνευση, αλλά να μπορείς να πάρεις σωστή απόφαση όταν το ίδιο ζήτημα εμφανιστεί μέσα σε πραγματική εφαρμογή.

## Το νοητικό μοντέλο

Ξεκίνα ορίζοντας τι εισέρχεται στη λειτουργία, ποιος έχει δικαίωμα να την ενεργοποιήσει, ποιος κανόνας εφαρμόζεται και ποιο αποτέλεσμα πρέπει να παραχθεί. Στο θέμα «Συναρτήσεις και παράμετροι» η καθαρή ευθύνη είναι σημαντικότερη από την ποσότητα κώδικα. Αν μία λειτουργία κάνει πολλά άσχετα πράγματα, διαίρεσέ την σε μικρότερα βήματα με σαφείς εισόδους και εξόδους.

## Πρακτικοί κανόνες

- Περιέγραψε πρώτα το αναμενόμενο αποτέλεσμα για το «Συναρτήσεις και παράμετροι».
- Ξεχώρισε δεδομένα, κανόνες, παρουσίαση και εξωτερικές εξαρτήσεις.
- Έλεγξε κενές τιμές, καθυστέρηση, διπλή ενέργεια και αποτυχία δικτύου.
- Μην εμπιστεύεσαι δεδομένα χρήστη πριν από validation και authorization.

## Μικρό παράδειγμα

Σε ένα e-shop, εφάρμοσε το «Συναρτήσεις και παράμετροι» στη ροή αγοράς: ο χρήστης επιλέγει προϊόν, το σύστημα επαληθεύει τα δεδομένα, εκτελεί τη λειτουργία μία φορά και επιστρέφει σαφή κατάσταση επιτυχίας ή αποτυχίας. Το αποτέλεσμα πρέπει να παραμένει προβλέψιμο ακόμη και αν ο χρήστης πατήσει δύο φορές ή η σύνδεση διακοπεί.

| **Έλεγχος κατανόησης** | Μπορείς να εξηγήσεις το «Συναρτήσεις και παράμετροι» χωρίς ορολογία, να δώσεις ένα παράδειγμα και να ονομάσεις δύο τρόπους αποτυχίας; |
|---|---|

## Prompt εφαρμογής

```
Εξέτασε το project μου ως προς: Συναρτήσεις και παράμετροι.
Δώσε: τρέχουσα ροή, κινδύνους, προτεινόμενη αλλαγή,
κριτήρια αποδοχής και ελάχιστες δοκιμές.
Μην αλλάξεις κώδικα πριν παρουσιάσεις το σχέδιο.
```

# 2.5 Arrays, objects και async ροές

Η ενότητα Arrays, objects και async ροές αποτελεί βασικό δομικό στοιχείο του κεφαλαίου «Βασικές αρχές κώδικα». Στόχος δεν είναι η μηχανική απομνημόνευση, αλλά να μπορείς να πάρεις σωστή απόφαση όταν το ίδιο ζήτημα εμφανιστεί μέσα σε πραγματική εφαρμογή.

## Το νοητικό μοντέλο

Ξεκίνα ορίζοντας τι εισέρχεται στη λειτουργία, ποιος έχει δικαίωμα να την ενεργοποιήσει, ποιος κανόνας εφαρμόζεται και ποιο αποτέλεσμα πρέπει να παραχθεί. Στο θέμα «Arrays, objects και async ροές» η καθαρή ευθύνη είναι σημαντικότερη από την ποσότητα κώδικα. Αν μία λειτουργία κάνει πολλά άσχετα πράγματα, διαίρεσέ την σε μικρότερα βήματα με σαφείς εισόδους και εξόδους.

## Πρακτικοί κανόνες

- Περιέγραψε πρώτα το αναμενόμενο αποτέλεσμα για το «Arrays, objects και async ροές».
- Ξεχώρισε δεδομένα, κανόνες, παρουσίαση και εξωτερικές εξαρτήσεις.
- Έλεγξε κενές τιμές, καθυστέρηση, διπλή ενέργεια και αποτυχία δικτύου.
- Μην εμπιστεύεσαι δεδομένα χρήστη πριν από validation και authorization.

## Μικρό παράδειγμα

Σε ένα e-shop, εφάρμοσε το «Arrays, objects και async ροές» στη ροή αγοράς: ο χρήστης επιλέγει προϊόν, το σύστημα επαληθεύει τα δεδομένα, εκτελεί τη λειτουργία μία φορά και επιστρέφει σαφή κατάσταση επιτυχίας ή αποτυχίας. Το αποτέλεσμα πρέπει να παραμένει προβλέψιμο ακόμη και αν ο χρήστης πατήσει δύο φορές ή η σύνδεση διακοπεί.

| **Έλεγχος κατανόησης** | Μπορείς να εξηγήσεις το «Arrays, objects και async ροές» χωρίς ορολογία, να δώσεις ένα παράδειγμα και να ονομάσεις δύο τρόπους αποτυχίας; |
|---|---|

## Prompt εφαρμογής

```
Εξέτασε το project μου ως προς: Arrays, objects και async ροές.
Δώσε: τρέχουσα ροή, κινδύνους, προτεινόμενη αλλαγή,
κριτήρια αποδοχής και ελάχιστες δοκιμές.
Μην αλλάξεις κώδικα πριν παρουσιάσεις το σχέδιο.
```

# 2.6 Εργαστήριο κεφαλαίου

Υλοποίησε τον υπολογισμό καλαθιού με έκπτωση και έλεγχο αποθέματος.

## Οδηγίες

- Γράψε το happy path σε 5–8 αριθμημένα βήματα.
- Κατάγραψε τουλάχιστον τέσσερις αποτυχίες ή ακραίες περιπτώσεις.
- Όρισε source of truth και ποιο μέρος έχει κάθε ευθύνη.
- Ζήτησε από το Codex σχέδιο και έλεγξέ το πριν επιτρέψεις αλλαγές.
- Υλοποίησε τη μικρότερη λειτουργική έκδοση και δοκίμασέ την.

## Παράδειγμα αναφοράς

```
const total = items.reduce((sum, item) =>
  sum + item.price * item.quantity, 0);
```

## Definition of Done

| | |
|---|---|
| Λειτουργία | Το βασικό σενάριο περνά από άκρη σε άκρη. |
| Αποτυχίες | Ο χρήστης λαμβάνει χρήσιμα μηνύματα χωρίς απώλεια δεδομένων. |
| Ασφάλεια | Secrets και αποφάσεις δικαιωμάτων μένουν στον server. |
| Δοκιμές | Υπάρχει επαναλήψιμος έλεγχος για τα κρίσιμα σημεία. |
| Τεκμηρίωση | Άλλος άνθρωπος μπορεί να καταλάβει τη ροή. |

# 2.7 Quiz και checklist

Απάντησε πρώτα χωρίς να κοιτάξεις πίσω.

**Τι πρόβλημα λύνει το «Μεταβλητές και τύποι τιμών»;**

.....

**Ποια δεδομένα θεωρούνται αξιόπιστα στο «Βασικές αρχές κώδικα»;**

.....

**Ποια ευθύνη ανήκει στον client και ποια στον server;**

.....

**Ποιες δύο αποτυχίες πρέπει οπωσδήποτε να δοκιμαστούν;**

.....

**Πώς αποδεικνύεις ότι η υλοποίηση λειτουργεί;**

.....

## Checklist

- Μπορώ να εξηγήσω τα πέντε βασικά θέματα του κεφαλαίου 2.
- Έκανα το εργαστήριο σε πραγματικό ή δοκιμαστικό project.
- Δοκίμασα τουλάχιστον μία αποτυχία, όχι μόνο το happy path.
- Έγραψα καθαρά κριτήρια αποδοχής.
- Ξέρω ποιο είναι το επόμενο μικρό βήμα.

| | |
|---|---|
| **Κανόνας προόδου** | Αν μπορείς να εξηγήσεις, να εφαρμόσεις και να ελέγξεις, προχώρα. Αν απλώς αναγνωρίζεις τους όρους, επανάλαβε το εργαστήριο. |