Έγγραφο απαιτήσεων λογισμικού (SRS)

ΠΡΟΣΑΡΜΟΓΗ ΤΟΥ ΑΝΤΙΣΤΟΙΧΟΥ ΕΓΓΡΑΦΟΥ ΤΟΥ ΠΡΟΤΥΠΟΥ ISO/IEC/IEEE 29148:2011

${TEXT, “Project Name”}

# Εισαγωγή

## 1.1 Εισαγωγή: σκοπός του λογισμικού

${PROJECT, "Basic Information2"}

## 1.2 Διεπαφές (interfaces)

### 1.2.1 Διεπαφές με εξωτερικά συστήματα

${ELEMENT, “Web Browser Component Information”, Component, Any, PreferredElement.Name="Web Browser", "Details2"}

${ELEMENT, “Front End Component Information”, Component, Any, PreferredElement.Name="Svelte - Vite / front-end", "Details2"}

${ELEMENT, “Back End Component Information”, Component, Any, PreferredElement.Name="Node.js / back-end", "Details2"}

${ELEMENT, “Database Component Information”, Component, Any, PreferredElement.Name="MongoDB Database", "Details2"}

${DIAGRAM, “IntelliQ Component Diagram”, ComponentDiagram, Any, PreferredDiagram.Name="IntelliQ Components", "Basic"}

${DIAGRAM, “IntelliQ Deployment Diagram”, DeploymentDiagram, Any, PreferredDiagram.Name="IntelliQ Deployment", "Basic"}

### 1.2.2 Διεπαφές με το χρήστη

Το λογισμικό έχει πλούσιες δυνατότητες επικοινωνίας με τον χρήστη, τις οποίες δείχνουμε με το Use Case Διάγραμμα και αναλυτικότερα με τα UML Class/API Διαγράμματα (διαχειριστή και υπόλοιπων χρηστών), αφού το μεγαλύτερο μέρος της επικοινωνίας   
χρήστη – λογισμικού στο σύστημά μας γίνεται μέσω αιτημάτων στα endpoints.

${DIAGRAM, “All Use Cases Diagram”, UseCaseDiagram, Any, PreferredDiagram.Name="All Use Cases", "Basic"}

${DIAGRAM, “API Admin Endpoints Diagram”, ClassDiagram, Any, PreferredDiagram.Name="Admin Endpoints", "Basic"}

${DIAGRAM, “API Functionality Endpoints Diagram”, ClassDiagram, Any, PreferredDiagram.Name="Functionality Endpoints", "Basic"}

Ωστόσο, η διεπαφή επικοινωνίας με τους χρήστες δεν καλύπτει όλες αυτές τις δυνατότητες μιας και είμαστε διμελής ομάδα. Οι περιπτώσεις χρήσεις εξυπηρετούνται μέσω άμεσων HTTP μηνυμάτων από τον χρήστη με εξαίρεση τα Endpoints που εμπεριέχονται στο Use Case που υλοποιήσαμε και θα δείξουμε παρακάτω.

# Αναφορές - πηγές πληροφοριών

Το backend είναι βασισμένο στην αποθήκευση δεδομένων σε mongoDB βάση, οπότε η κατανόησή του προϋποθέτει εξοικείωση με αυτό το «οικοσύστημα» καθώς και γνώσεις συνηθισμένων εργαλείων σχετικά με την λειτουργία backend εφαρμογών όπως διαχείριση JSON δεδομένων και κάποιας κατάλληλης γλώσσας προγραμματισμού (προτείνεται JavaScript που είναι και η δική μας επιλογή). Σχετικά θέματα που έχουν ενδιαφέρον είναι το middleware σε εφαρμογές, η επικύρωση εισερχόμενων δεδομένων (validation) και η έννοια του REST API.

Η κατανόηση της επικοινωνίας με την χρήστη μέσω frontend χρειάζεται βασικές γνώσεις χρήσης κάποιου κατάλληλου framework (εμείς διαλέξαμε το svelte, που είναι αρκετά εύχρηστο και «ελαφρύ»), HTML και CSS.

Παραθέτουμε μερικές πηγές για τα παραπάνω θέματα:

* [MongoDB Documentation](https://www.mongodb.com/docs/)
* [Working with JSON - Learn web development | MDN (mozilla.org)](https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/JSON)
* [Using Express middleware (expressjs.com)](https://expressjs.com/en/guide/using-middleware.html)
* [What is REST - REST API Tutorial (restfulapi.net)](https://restfulapi.net/)
* [Docs • Svelte](https://svelte.dev/docs)
* [HTML: HyperText Markup Language | MDN (mozilla.org)](https://developer.mozilla.org/en-US/docs/Web/HTML)

# Προδιαγραφές απαιτήσεων λογισμικού

## 3.1 Περιπτώσεις χρήσης

 Υλοποιούμε μία περίπτωση χρήσης – μέσω της υλοποίησης της frontend εφαρμογής μας.

### 3.1.1 ΠΕΡΙΠΤΩΣΗ ΧΡΗΣΗΣ 1: ${DIAGRAM, “My First Use Case”, UseCaseDiagram, Any, PreferredDiagram.Name="Answer Questionnaire", "Basic2"}

#### 3.1.1.1 Χρήστες (ρόλοι) που εμπλέκονται

${ELEMENT, “Users for First Use Case”, Actor, Any, PreferredElement.Name="Απαντών", "Basic2"}

#### 3.1.1.2 Προϋποθέσεις εκτέλεσης

Η σύνδεση του χρήστη σε Web Browser και η πρόσβασή του στο URL της frontend εφαρμογής μας ή, αν ένας χρήστης επιθυμεί να απαντά άμεσα στις ερωτήσεις ενός ερωτηματολογίου, αρκεί να στέλνει μηνύματα HTTP στα αντίστοιχα End Points.

#### 3.1.1.3 Περιβάλλον εκτέλεσης

Η συγκεκριμένη περίπτωση χρήσης απαιτεί την διαθεσιμότητα όλων των κόμβων του συστήματός μας και για αυτό την διαλέξαμε. Αναφέρουμε πληροφορίες σχετικά με τους κόμβους που δείξαμε στο Deployment Διάγραμμα ώστε να καταστούν πιο σαφείς οι συνθήκες επιτυχούς εκτέλεσης του Use Case – Απάντηση σε Ερωτηματολόγιο.

${ELEMENT, “User Device Node Element”, Node, Any, PreferredElement.Name="User Device", "Basic"}

${ELEMENT, “Front-End Web Server Node Element”, Node, Any, PreferredElement.Name="Front-End Web Server", "Basic"}

${ELEMENT, “Back-End Server Node Element”, Node, Any, PreferredElement.Name="Back-End Server", "Basic"}

${ELEMENT, “Database Server Node Element”, Node, Any, PreferredElement.Name="Database Server", "Basic"}

#### 3.1.1.4 Δεδομένα εισόδου - εξόδου

##### 3.1.1.4.1 Δεδομένα εισόδου

Ως δεδομένα εισόδου θεωρούνται οτιδήποτε υποβάλει ο χρήστης προς το σύστημα στη προσπάθειά του να απαντήσει ένα ερωτηματολόγιο.

${ELEMENT, “Use Case Input Data”, "Class", LoopInDiagram, PreferredDiagram.Name="Request Bodies", Basic, PreferredExcludedInstances="j5FJBKGGAqACA4ct:e7OqBKGGAqAEBQ\_0, j5FJBKGGAqACA4ct:KWh5KmGGAqAEA6Rl, j5FJBKGGAqACA4ct:ttZlKmGGAqAEA6XJ, j5FJBKGGAqACA4ct:qPZhOKGGAqACA7JP, j5FJBKGGAqACA4ct:emJRKmGGAqAEA4CB"}

Επειδή η είσοδος γίνεται μέσω της εφαρμογής η μόνη εσφαλμένη επιλογή που μπορεί να δώσει ο χρήστης είναι λάθος τιμή αναγνωριστικού ερωτηματολογίου και αναγνωρίζεται από το query στη βάση δεδομένων το οποίο θα επιστρέψει μήνυμα λάθους.

##### 3.1.1.4.2 Δεδομένα εξόδου

Ως δεδομένα εξόδου θεωρούμε οτιδήποτε μπορεί να μας επιστρέψει η εφαρμογή στα παραπάνω αιτήματα εισόδου.

${ELEMENT, “Use Case Output Data”, "Class", LoopInDiagram, PreferredDiagram.Name="Response Bodies", Basic, PreferredExcludedInstances="d8VJBKGGAqACA4c6:I48z.KGGAqAEBQ1T, d8VJBKGGAqACA4c6:I48z.KGGAqAEBQ1T:cKez.KGGAqAEBQ1., d8VJBKGGAqACA4c6:I48z.KGGAqAEBQ1T:zpRz.KGGAqAEBQ2Q, d8VJBKGGAqACA4c6:I48z.KGGAqAEBQ1T:SCN6SmGGAqAEAyni, d8VJBKGGAqACA4c6:6VNxKmGGAqAEA4Gv, d8VJBKGGAqACA4c6:DafqBKGGAqAEBRBB, d8VJBKGGAqACA4c6:DafqBKGGAqAEBRBB:gMj7BKGGAqACA.Gc, d8VJBKGGAqACA4c6:Mp3TBKGGAqACA7V5:UWCbBKGGAqACA9.\_, d8VJBKGGAqACA4c6:f\_VGBKGGAqAEBRH2:Z7nLBKGGAqACA7Z2, d8VJBKGGAqACA4c6:lQx1KmGGAqAEA70d, d8VJBKGGAqACA4c6:3hv5KmGGAqAEA6TZ, d8VJBKGGAqACA4c6:.FA8BKGGAqAEBQ6x, d8VJBKGGAqACA4c6:.FA8BKGGAqAEBQ6x:ZDZyBKGGAqAEBQ7g"}

#### 3.1.1.5 Αλληλουχία ενεργειών - επιθυμητή συμπεριφορά

Δείχνουμε αρχικά, τα Sequence Διαγράμματα που εμπεριέχουν όλες τις δυνατές ενέργειες.

${DIAGRAM, “Answer Questionnaire Diagram”, InteractionDiagram, Any, PreferredDiagram.Name="Answer Questionnaire Sequence", "Basic"}

${DIAGRAM, “Fetch Data Diagram”, InteractionDiagram, Any, PreferredDiagram.Name="Fetch Data", "Basic"}

${DIAGRAM, “Answer a Question Diagram”, InteractionDiagram, Any, PreferredDiagram.Name="Answer a Question", "Basic"}

Και περιγράφουμε τα μηνύματα που ανταλλάσσονται μεταξύ των οντοτήτων στα διαγράμματα αυτά.

${DIAGRAM, “Answer Questionnaire Messages”, InteractionDiagram, Any, PreferredDiagram.Name="Answer Questionnaire Sequence", "Details2"}

${DIAGRAM, “Fetch Data Messages”, InteractionDiagram, Any, PreferredDiagram.Name="Fetch Data", "Details2"}

${DIAGRAM, “Answer a Question Messages”, InteractionDiagram, Any, PreferredDiagram.Name="Answer a Question", "Details2"}

Η βασική λειτουργία μπορεί να περιγραφεί ως εξής:

1. Ο χρήστης δίνει ένα questionnaireID και αναμένει να λάβει την πρώτη ερώτηση αυτού του ερωτηματολογίου.
2. Η ερώτηση προβάλλεται στον χρήστη με κατάλληλη μορφή αναλόγως αν είναι open string ερώτηση ή/και πρέπει να απαντηθεί αναγκαστικά.
3. Ο χρήστης επιλέγει μια απάντηση ή τη συμπληρώνει μόνος του.
4. Υποβάλει την απάντησή του πατώντας το Submit κουμπί
   1. Προαιρετικό βήμα: Αν είναι η πρώτη απάντηση που υποβάλλεται, δημιούργησε ένα αναγνωριστικό συνόδου (session ID) για αυτό το γεγονός απάντησης, το οποίο θα χρησιμοποιήσεις και στις απαντήσεις σου στις υπόλοιπες ερωτήσεις του ερωτηματολογίου.
5. Αναμένει την προβολή της επόμενης ερώτησης.
6. Συνεχίζει από το 2. Μέχρι να μην υπάρχουν άλλες ερωτήσεις.
7. Τέλος, του προβάλλεται μια περίληψη του γεγονότος απαντήσεων που μόλις ολοκλήρωσε.

Ο χρήστης έχει πάντοτε διαθέσιμο τα Clear και Cancel κουμπιά. Τα οποία καθαρίζουν την επιλογή του στη τωρινή ερώτηση και ακυρώνουν όλη τη σύνοδο πηγαίνοντας στο βήμα 1 αντίστοιχα.

Στην οθόνη του βήματος 7 έχει διαθέσιμο το κουμπί Reset που τον επαναφέρει στο βήμα 1.

Σε περίπτωση σφάλματος (λάθος αίτημα ερωτηματολογίου ή πρόβλημα στη λειτουργία της εφαρμογής) ο χρήστης ειδοποιείται μέσω ειδικής οθόνης η οποία έχει Reset επιλογή που τον επαναφέρει στο βήμα 1.

Σχετικά με το βήμα 4a: η frontend εφαρμογή μας δοκιμάζει να δημιουργήσει αναγνωριστικό συνόδου 3 φορές σε περίπτωση που το μήνυμα που λάβει από το backend δίνει την εντύπωση ότι το αναγνωριστικό που δημιούργησε αντιστοιχεί σε ήδη αποθηκευμένη σύνοδο (που έχει γίνει στο παρελθόν από τον ίδιο ή άλλο χρήστη). Προφανώς, όλες οι απαντήσεις που αναφέρονται σε ένα ερωτηματολόγιο και κατά τη διάρκεια ενός γεγονότος απάντησης αναφέρουν το ίδιο αναγνωριστικό συνόδου.

Σχηματικά, βλέπουμε τα εξής:

${DIAGRAM, “Activity Diagram of our Use Case”, ActivityDiagram, Any, PreferredDiagram.Name="Answer Questionnaire", "Basic"}

#### 3.1.1.7 Δεδομένα εξόδου

Τα δεδομένα εξόδου εκτός από τα προηγούμενα αντικείμενα που αφορούσαν την πλευρά του χρήστη περιλαμβάνουν και τα αντικείμενα απάντησης που αποθηκεύονται από κάθε υποβολή απάντησης. Το πώς/πότε παράγονται φαίνεται στα διαγράμματα (Fetch Data διάγραμμα το οποίο έχεις ως δεδομένα την απάντηση του χρήστη και είναι εμφανές Answer Questionnaire διάγραμμα). Περιγράφουμε και αυτό το αντικείμενο:

${ELEMENT, “Answer Object”, Class, Any, PreferredElement.Name="Answer", "minimalDetails"}

#### 3.1.1.8 Παρατηρήσεις

Η frontend εφαρμογή παρουσιάζει στο τέλος και περίληψη των απαντήσεων του χρήστη στο ερωτηματολόγιο που προηγήθηκε, οπότε υλοποιεί σε μερικό βαθμό και το “ ${ELEMENT, “Second Use Case Title”, UseCase, Any, PreferredElement.Name="View Own Session Answers", "UseCaseTitle"}” Use Case.

## 3.2 Απαιτήσεις επιδόσεων

Το λογισμικό πρέπει να ανταποκρίνεται στους χρήστες και να δίνει κατάλληλα μηνύματα σε κάθε περίπτωση λάθους εντός εύλογου χρονικού διαστήματος. Η διάταξή του πρέπει να είναι εύληπτη και εύχρηστη. Το backend έχει σχεδιαστεί ώστε να επιστρέφει μηνύματα λαθών που αφορούν τη βάση και το frontend εμφανίζει τα μηνύματα αυτά αλλά και μηνύματα γενικής αποτυχίας, όπως μη διαθεσιμότητα του backend γενικά.

Έχουν χρησιμοποιηθεί αρκετά «ελαφριά» εργαλεία για την δημιουργία του και είναι πρόγραμμα που δεν απαιτεί ιδιαίτερους υπολογιστικούς πόρους, οπότε υποστηρίζονται αρκετοί πολλαπλοί χρήστες – αναλόγως τις δυνατότητες του hosting server να τους εξυπηρετήσει. Προς το σκοπό αυτό έχουμε προσπαθήσει και στο backend και στο frontend να αξιοποιήσουμε όσο το δυνατό περισσότερο τις δυνατότητες ασύγχρονης λειτουργίας της JavaScript.

Το λογισμικό είναι κλιμακώσιμο καθώς το API του έχει σχεδιαστεί ώστε να υπακούει στις αρχές του RESTful API και μπορεί να επεκταθεί με middleware. Επίσης, αν δεν επιθυμούμε παρέμβαση στον πηγαίο κώδικα, πιθανό bottleneck ίσως αποτελεί η βάση δεδομένων, ωστόσο, και αυτή μπορεί να αντικατασταθεί με ό,τι κρίνει ο χρήστης κατάλληλο. Αρκεί να υποστηρίζει interface παρόμοιο με της MongoDB.

Το λογισμικό παρουσιάζει συμβατότητα με πληθώρα συσκευών, αφού μοναδική του απαίτηση είναι η πρόσβαση στον διαδικτυακό σύνδεσμο του frontend. Αρκεί ο χρήστης να έχει πρόσβαση σε έναν Web Browser.

Το backend έχει τη δυνατότητα να επιστρέφει πληθώρα μεταδεδομένων για τις πληροφορίες που επεξεργάζεται, οπότε βοηθάει τον χρήστη στο να διαπιστώσει πιθανά bottlenecks και γενικά να κατανοήσει τη πραγματικού – χρόνου συμπεριφορά του.

Από testing που πραγματοποιήσαμε όσον αφορά τη χρήση του συνολικού συστήματος (μέσω του frontend ενώ συνδέεται με το υπόλοιπο λογισμικό όπως περιγράφει το Deployment Diagram) μάλλον η περίπτωση που παρουσιάζει μεγαλύτερη της επιθυμητής καθυστέρηση είναι η αποστολή αιτήματος από το frontend ενώ το backend είναι εκτός λειτουργίας. Βέβαια, ο λόγος είναι ότι το frontend περιμένει ένα εύλογο χρονικό διάστημα μέχρι να διαπιστώσει ότι το backend δεν ανταποκρίνεται.

## 3.3 Απαιτήσεις οργάνωσης δεδομένων

### 3.3.1 Απαιτήσεις και περιορισμοί πρόσβασης σε δεδομένα

Δείχνουμε τις δομές δεδομένων που χρησιμοποιεί το λογισμικό μας.

${DIAGRAM, “Database Schema”, ClassDiagram, Any, PreferredDiagram.Name="Database Schema", "Basic"}

Και οι δομές των αιτημάτων (αριστερά βρίσκονται αυτές που αναφέρονται σε λειτουργικά endpoints και δεξιά αυτές που αναφέρονται σε διαχειριστικά endpoints).

${DIAGRAM, “Endpoint Request Bodies”, ClassDiagram, Any, PreferredDiagram.Name="Request Bodies", "Basic"}

${DIAGRAM, “Endpoint Response Bodies”, ClassDiagram, Any, PreferredDiagram.Name="Response Bodies", "Basic"}

Δεν έχουν υλοποιηθεί δυνατότητες ασφάλειας/διαχωρισμού διαχειριστών από τους υπόλοιπους χρήστες, οπότε οποιοσδήποτε μπορεί να λειτουργήσει ως κάποιος από τους δύο ρόλους υποβάλλοντας HTTP requests στο API μας. Το frontend απευθύνεται μόνο σε χρήστες – απαντώντες στα ερωτηματολόγια.

Οι παραπάνω δομές πρέπει να είναι μοντελοποιημένες ως JSON (Content-Type: application/json) με εξαίρεση τα Response Bodies στη περίπτωση επιτυχίας λειτουργικών endpoints που μπορεί να είναι csv αν ο χρήστης συμπεριλάβει την προαιρετική query παράμετρο “format” με τιμή csv (Content-Type: text/csv).

## 3.5 Λοιπές απαιτήσεις

### 3.5.1 Απαιτήσεις διαθεσιμότητας λογισμικού

Η εφαρμογή είναι επιθυμητό να είναι διαθέσιμη καθημερινά για όλους τους χρήστες μας. Συνεπώς, οι εξυπηρετητές που τρέχουν το Frontend Application, το Backend και τη βάση πρέπει να υποστηρίζουν αυτή τη λειτουργία.

Επίσης, η εφαρμογή πρέπει να τηρεί συνήθη standard ανταλλαγής HTTP(S) μηνυμάτων μέσω διαδικτύου για να συμφωνεί με οποιονδήποτε ISP μπορεί να έχει κάποιος χρήστης μας.

### 3.5.2 Απαιτήσεις ασφάλειας

Δεν έχει υλοποιηθεί login ή/και προστασία των δεδομένων. Η ασφάλεια στηρίζεται στο ήθος των χρηστών μας.

Το frontend επικοινωνεί με τους User Browsers μέσω https ως μια μικρή απόπειρα παροχής υπηρεσιών ασφαλείας.