Ιωάννης Α. Μπούχρας

ioabuh@unixfor.gr

Περιγραφή

Το παρόν έγγραφο περιγράφει τις τεχνικές προδιαγραφές για την ανάπτυξη ενός λογισμικού μετατροπής email σε αιτήματα υποστήριξης (tickets), με δύο modes λειτουργίας: Classic (με κανόνες parsing) και AI (με χρήση GPT-4 Vision). Το σύστημα υποστηρίζει πρωτόκολλα IMAP/POP3, αποθήκευση email και συνημμένων σε βάση, εξαγωγή πεδίων ανά πελάτη, και επικοινωνία με εξωτερικά συστήματα μέσω API. Προσφέρει παραμετροποίηση, logging, και επεκτασιμότητα.

**Τεχνικές Προδιαγραφές Συστήματος "Email to Ticket"**

περιεχόμενα

[Εισαγωγή 2](#_Toc198023825)

[Αρχιτεκτονική & Κοινή Υποδομή 2](#_Toc198023826)

[Λειτουργία Classic Mode 2](#_Toc198023827)

[1. Σύνδεση σε Mail Server (POP3/IMAP) 2](#_Toc198023828)

[2. Ανάκτηση & Αποθήκευση Emails σε Ουρά (Queue DB) 3](#_Toc198023829)

[3. Κανόνες Εξαγωγής Δεδομένων (Parsing & Extraction Rules) 3](#_Toc198023830)

[4. Εισαγωγή σε Κανονικοποιημένο Πίνακα Tickets 4](#_Toc198023831)

[5. Σήμανση Email ως Processed 5](#_Toc198023832)

[6. Logging (Classic Mode) 5](#_Toc198023833)

[7. Επεκτασιμότητα & Modular Υποδομή (Classic Mode) 6](#_Toc198023834)

[Λειτουργία AI Mode 6](#_Toc198023835)

[1. Σύνδεση & Λήψη Emails (ίδιο με Classic) 7](#_Toc198023836)

[2. Επεξεργασία Περιεχομένου με AI (GPT-4 Vision) 7](#_Toc198023837)

[3. Εξαγωγή Πεδίων & Καταχώριση Ticket (AI Mode) 8](#_Toc198023838)

[4. Σήμανση Email ως Processed (AI Mode) 9](#_Toc198023839)

[5. Logging (AI Mode) 9](#_Toc198023840)

[6. Κοινή Υποδομή & Εναλλαγή Λειτουργίας 10](#_Toc198023841)

[Κεντρική Παραμετροποίηση Συστήματος 10](#_Toc198023842)

[API Endpoint για Τελικά Συστήματα Ticketing 12](#_Toc198023843)

[1. Σκοπός & Λειτουργικότητα API 12](#_Toc198023844)

[2. Διαθέσιμα Endpoint & Λειτουργίες 12](#_Toc198023845)

[3. Ενέργειες μετά την Ανάκτηση 13](#_Toc198023846)

[4. Rate Limiting & Performance 13](#_Toc198023847)

[Απαιτήσεις Λογισμικού & Εργαλεία Υλοποίησης 14](#_Toc198023848)

[Συμπέρασμα 15](#_Toc198023849)

## Εισαγωγή

Το σύστημα "Email to Ticket" είναι μια υπηρεσία (Windows Service ή Services) σε υποστηριζόμενα συστήματα (Windows & Linux Debian based) γραμμένο στην γλώσσα προγραμματισμού C# που μετατρέπει εισερχόμενα emails σε εγγραφές ticket (αιτήματα) σε βάση δεδομένων. Παρέχει δύο modes λειτουργίας – **Classic Mode** και **AI Mode** – επιτρέποντας είτε κλασική εξαγωγή δεδομένων βάσει κανόνων είτε προηγμένη ανάλυση περιεχομένου με τη βοήθεια τεχνητής νοημοσύνης. Το σύστημα υποστηρίζει πολλαπλούς πελάτες (configurations) και πολλαπλά τελικά συστήματα διαχείρισης tickets, με κοινή υποδομή ουράς (queue), logging και API πρόσβασης. Σκοπός των παρακάτω προδιαγραφών είναι να καθοδηγήσουν την υλοποίηση, διασφαλίζοντας επεκτασιμότητα, αξιοπιστία και ευκολία παραμετροποίησης για τους C# developers.

## Αρχιτεκτονική & Κοινή Υποδομή

Το σύστημα αποτελείται από διακριτά υποσυστήματα που συνεργάζονται αρμονικά:

* **Mail Retrieval Service**: Υπεύθυνο για τη σύνδεση στον mail server (IMAP/POP3) και την ανάκτηση των νέων emails, ανεξαρτήτως mode. Χρησιμοποιεί ασφαλείς συνδέσεις (SSL/TLS) όπου είναι διαθέσιμες.
* **Queue Database**: Κοινή ουρά αποθήκευσης εισερχόμενων emails. Κάθε νέο email (μαζί με τα συνημμένα του) εισάγεται προσωρινά σε πίνακα "EmailQueue" στη βάση δεδομένων (υποστηρίζονται MS SQL, MySQL, PostgreSQL).
* **Processing Engine**: Ανάλογα με το επιλεγμένο mode (Classic ή AI), εφαρμόζει αντίστοιχη μέθοδο επεξεργασίας για να εξάγει τα απαιτούμενα πεδία και να δημιουργήσει ένα ticket.
* **Normalized Ticket Database**: Τελικός πίνακας καταχώρισης tickets, με ενοποιημένα πεδία (π.χ. Αριθμός Αιτήματος, Πελάτης, Θέμα, Περιγραφή, Προτεραιότητα, Συνημμένα κλπ).
* **Integration/API Module**: Παρέχει διεπαφή (API endpoint) ώστε εξωτερικά ή/και πολλαπλά συστήματα ticketing να μπορούν να ανακτούν τα νέα tickets από το σύστημα. Επίσης, υποστηρίζει modular connectors για άμεση ενσωμάτωση σε διάφορα τελικά συστήματα.
* **Logging Service**: Κεντρικό σύστημα καταγραφής που συλλέγει λεπτομερώς γεγονότα από όλα τα στάδια (σύνδεση mail, parsing, εξαγωγή δεδομένων, κ.λπ.), προσφέροντας δυνατότητες debugging και monitoring.

Η σχεδίαση είναι **modular**, διευκολύνοντας την εναλλαγή μεταξύ Classic και AI mode χωρίς αλλαγές στην υπόλοιπη υποδομή. Οι κοινές λειτουργίες (π.χ. ανάγνωση email, αποθήκευση σε queue, εισαγωγή σε πίνακα tickets, logging) υλοποιούνται μία φορά και παραμετροποιούνται ανά mode μέσω κεντρικού configuration.

## Λειτουργία Classic Mode

Στην κλασική λειτουργία, το σύστημα ακολουθεί έναν κανονικό γραμμή (pipeline) επεξεργασίας email με κανόνες. Το Classic Mode δίνει έμφαση σε προκαθορισμένους κανόνες ανάλυσης κειμένου και απλούστερες διαδικασίες, χωρίς χρήση AI. Περιλαμβάνει τα εξής στάδια:

### 1. Σύνδεση σε Mail Server (POP3/IMAP)

* **Υποστήριξη Πρωτοκόλλων**: Η υπηρεσία υποστηρίζει σύνδεση σε mail servers μέσω των πρωτοκόλλων **POP3** και **IMAP**. Χρησιμοποιείται .NET βιβλιοθήκη (π.χ. MailKit) για τη διαχείριση των email protocols, που προσφέρει πλήρεις δυνατότητες IMAP/POP3 (ανάγνωση, λήψη emails, διαχείριση φακέλων).
* **Ασφάλεια Σύνδεσης**: Υποστηρίζεται **κρυπτογραφημένη επικοινωνία SSL/TLS** (όπου ο mail server το παρέχει) καθώς και απλή μη κρυπτογραφημένη σύνδεση. Η βιβλιοθήκη MailKit, για παράδειγμα, επιτρέπει εύκολη ενεργοποίηση TLS τόσο σε IMAP όσο και σε POP. Το σύστημα πρέπει να προσπαθεί πρώτα με ασφαλή σύνδεση (SSL/TLS) και να υποβιβάζει σε απλή μόνο αν χρειάζεται (βάσει ρυθμίσεων).
* **Διαπιστεύσεις**: Οι λογαριασμοί email (διεύθυνση, username, password) και οι ρυθμίσεις διακομιστή (host, port, χρήση TLS) καθορίζονται στο κεντρικό configuration. Η αυθεντικοποίηση γίνεται με ασφαλή τρόπο (π.χ. μέσω OAuth2 όπου είναι εφικτό ή αποθήκευση κρυπτογραφημένων credentials).
* **Πολυπλοκότητα & Αξιοπιστία**: Το σύστημα επιτρέπει τη ρύθμιση πολλαπλών λογαριασμών email/θυρίδων προς έλεγχο. Η ανάγνωση των emails μπορεί να υλοποιηθεί περιοδικά (polling κάθε Χ λεπτά) ή σε πραγματικό χρόνο μέσω IMAP IDLE. Σημείωση: Το MailKit υποστηρίζει την εντολή IMAP **IDLE** για ειδοποίηση νέων emails σε πραγματικό χρόνο, παρέχοντας δυνατότητα άμεσης αντίδρασης σε νέα μηνύματα (εφόσον το επιτρέπει ο IMAP server). Για POP3 (που δεν έχει push notifications), χρησιμοποιείται χρονοδρομολόγηση (interval polling).

### 2. Ανάκτηση & Αποθήκευση Emails σε Ουρά (Queue DB)

* **Λήψη Μηνυμάτων**: Μετά τη σύνδεση, το σύστημα **σαρώνει τα νέα εισερχόμενα emails**. Στην περίπτωση IMAP, θα αναζητά μηνύματα που δεν έχουν σημανθεί ως “seen” (ή σε ειδικό φάκελο Inbox). Στην περίπτωση POP3, θα λαμβάνει όλα τα μηνύματα (πιθανόν διαγράφοντας τα από τον server μετά την επιτυχή λήψη, αν αυτό οριστεί).
* **Δομή Ουράς**: Κάθε email εισάγεται σε έναν πίνακα **EmailQueue** στη βάση δεδομένων. Αυτός λειτουργεί ως προσωρινή αποθήκη/ουρά επεξεργασίας πριν τη δημιουργία ticket. Η εγγραφή περιλαμβάνει μεταξύ άλλων πεδία:
  + **QueueID** (μοναδικό αναγνωριστικό εγγραφής).
  + **Sender**, **Recipients** (Αποστολέας, Παραλήπτες – διευθύνσεις email).
  + **Subject** (Θέμα email).
  + **BodyType** (τύπος σώματος, π.χ. text/plain ή text/html).
  + **BodyContent** (το πλήρες σώμα του email σε κείμενο ή HTML μορφή – διατηρείται αυτούσιο).
  + **ReceivedDate** (ημ/νία λήψης).
  + **Attachments**: Λίστα συνημμένων αρχείων. Για κάθε συνημμένο αποθηκεύονται μεταδεδομένα όπως Όνομα Αρχείου, Τύπος MIME, Μέγεθος, καθώς και το περιεχόμενο. Το περιεχόμενο μπορεί είτε να αποθηκεύεται ως binary (BLOB) στη βάση, είτε σε διαμοιραζόμενο αποθηκευτικό χώρο με αναφορά διαδρομής. Υποστηρίζονται αρχεία τύπου PDF, DOCX (Word), TXT, JPG, PNG κ.ά.
* **Συλλογή Συνημμένων**: Κατά την εισαγωγή στην ουρά (queue), το σύστημα εξάγει όλα τα attachments του email. Μέσω της βιβλιοθήκης email, γίνεται έλεγχος των body parts του μηνύματος για συνημμένα και αποθήκευση τους στην παραπάνω δομή. (Π.χ. με MailKit, επανάληψη σε message.Attachments και DecodeTo αποθήκευση αρχείου) Όλα τα συνημμένα συνδέονται με την εγγραφή του email στην ουρά (σχέση 1-προς-Ν).
* **Σήμανση στην Ουρά**: Κάθε νέα εγγραφή στην EmailQueue φέρει ένα πεδίο Processed (ή status) με αρχική τιμή "Pending". Αυτό το status θα ενημερωθεί όταν ολοκληρωθεί η επεξεργασία (δημιουργία ticket) για να αποφευχθεί διπλή επεξεργασία.

### 3. Κανόνες Εξαγωγής Δεδομένων (Parsing & Extraction Rules)

* **Ρόλος Κανόνων**: Αφού το email εισαχθεί στην ουρά, ενεργοποιείται η μηχανή κανόνων εξαγωγής για Classic Mode. Πρόκειται για ένα σύνολο από προκαθορισμένους ή/και παραμετροποιήσιμους κανόνες ανά πελάτη, που καθορίζουν πώς θα εξαχθούν συγκεκριμένα πεδία από το περιεχόμενο του email ή/και των συνημμένων.
* **Παραμετροποίηση ανά Πελάτη**: Οι κανόνες ορίζονται ανά πελάτη/εγκατάσταση (π.χ. σε αρχείο ρυθμίσεων ή πίνακα DB). Για κάθε πελάτη μπορούν να καθοριστούν:
  + Ποια πεδία ticket πρέπει να εξαχθούν (π.χ. "Τίτλος", "Περιγραφή", "Όνομα Πελάτη", "Αριθμός Σύμβασης" κλπ).
  + Κανόνες ή patterns για κάθε πεδίο, που εφαρμόζονται στο σώμα email ή σε συγκεκριμένο συνημμένο.
* **Μορφή Κανόνων**: Οι κανόνες μπορεί να είναι με τη μορφή regex patterns, σταθερών δεικτών (keywords) ή θέση σε φορμά. Παράδειγμα: Ένας κανόνας regex για πεδίο "Αριθμός Αιτήματος" μπορεί να οριστεί ως Ticket ID: (\d+) ώστε να αντλεί τον αριθμό αμέσως μετά τη φράση "Ticket ID:". Άλλος κανόνας μπορεί να ψάχνει στο σώμα για μια γραμμή που ξεκινά με "Όνομα Πελάτη:" και να αποθηκεύει το υπόλοιπο ως τιμή.
* **Εξαγωγή από Συνημμένα**: Το σύστημα υποστηρίζει και parsing συνημμένων εφόσον τα δεδομένα βρίσκονται εκεί. Αν για παράδειγμα ο πελάτης στέλνει πάντα μια φόρμα σε PDF, μπορεί να ενσωματωθεί διαδικασία ανάγνωσης PDF (π.χ. με χρήση βιβλιοθήκης iTextSharp ή PdfSharp για εξαγωγή κειμένου) και στη συνέχεια εφαρμογή των κανόνων πάνω στο κείμενο του PDF. Παρόμοια, για αρχεία Word (DOCX) μπορεί να χρησιμοποιηθεί το OpenXML SDK ή Apache POI για ανάγνωση περιεχομένου. Οι κανόνες μπορούν να προσδιορίζουν συγκεκριμένο όνομα αρχείου ή τύπο MIME όπου θα ψάξουν, π.χ. "Αν υπάρχει συνημμένο PDF με όνομα Invoice, εξήγαγε αριθμό παραγγελίας από εκεί".
* **Μηχανισμός Κανόνων**: Η υλοποίηση μπορεί να γίνει μέσω **Rule Engine** ή απλού processing pipeline:
  + Διατρέχει τους ορισμένους κανόνες κατά σειρά προτεραιότητας.
  + Για κάθε κανόνα, προσπαθεί να βρει τα αντίστοιχα δεδομένα. Αν αποτύχει και το πεδίο είναι υποχρεωτικό, καταγράφεται σφάλμα ή το ticket μαρκάρεται ως incompletely parsed.
  + Υποστηρίζεται μετασχηματισμός δεδομένων αν χρειάζεται (π.χ. μετατροπή ημερομηνιών σε ενιαία μορφή, αριθμών σε συγκεκριμένο format).
* **Παράδειγμα**: Για έναν πελάτη Α, μπορεί να υπάρχει κανόνας:
  + Field "Title": Να τεθεί το Subject του email.
  + Field "Description": Αν το email είναι HTML, να εξαχθεί μόνο το κείμενο (stripping HTML tags) για καθαρή περιγραφή.
  + Field "Priority": Αναζήτησε λέξεις "High/Medium/Low" στο σώμα ή θέμα και αντιστοίχισε σε τιμή προτεραιότητας.
  + Field "CustomerID": Έλεγξε συνημμένο PDF για γραμμή "Customer ID:".
* **Αποτυχία Κανόνων**: Αν κάποιο απαιτούμενο πεδίο δεν αντληθεί λόγω format που δεν ταιριάζει με τους κανόνες, το σύστημα μπορεί:
  + είτε να δημιουργεί ticket με κενό πεδίο και να το σημάνει για χειροκίνητο έλεγχο,
  + είτε να μαρκάρει το email ως μη επεξεργάσιμο (error state) και να προχωρά στο επόμενο, με κατάλληλη ειδοποίηση (logging/alert).

### 4. Εισαγωγή σε Κανονικοποιημένο Πίνακα Tickets

* **Δομή Πίνακα Tickets**: Η βάση δεδομένων περιλαμβάνει έναν πίνακα Tickets (ή παρόμοιο όνομα) όπου καταχωρούνται τα αποτελέσματα επεξεργασίας κάθε email. Αυτός ο πίνακας έχει **normalized σχήμα**, δηλαδή σταθερές στήλες που καλύπτουν όλα τα απαραίτητα πεδία ενός αιτήματος, ανεξαρτήτως πελάτη. Ενδεικτικά πεδία:
  + TicketID (πρωτεύον κλειδί).
  + ClientID ή SystemID (αντιστοιχεί στον πελάτη ή στο τελικό σύστημα που ανήκει το αίτημα).
  + Title (Σύντομος τίτλος/θέμα αιτήματος).
  + Description (Αναλυτική περιγραφή).
  + CustomerName, ContractNumber, και άλλα πεδία που έχουν οριστεί από τους κανόνες (προσαρμόζεται το schema ώστε να περιλαμβάνει κοινά πεδία όλων των πελατών ή χρησιμοποιείται επέκταση όπως extra JSON field για πολύ εξειδικευμένα δεδομένα).
  + Attachments (σύνδεση προς πίνακα TicketAttachments όπου αποθηκεύονται τυχόν συνημμένα αρχεία σχετιζόμενα με το ticket).
  + CreatedDate (ημερομηνία δημιουργίας του ticket).
  + SourceEmailID (foreign key προς το EmailQueue ή απευθείας το MessageID του email για ίχνος).
* **Καταχώριση Δεδομένων**: Μετά την επιτυχή εξαγωγή μέσω των κανόνων, η εφαρμογή δημιουργεί μια νέα εγγραφή στον πίνακα Tickets. Γίνονται οι κατάλληλες μετατροπές τύπων δεδομένων (π.χ. string σε int όπου απαιτείται) και εισαγωγή όλων των διαθέσιμων πεδίων.
* **Συσχέτιση Συνημμένων**: Αν υπάρχουν συνημμένα που σχετίζονται με το ticket, εγγράφονται στον πίνακα TicketAttachments (ή ισοδύναμο) με αναφορά στο TicketID. Στην Classic mode, τυπικά θα αντιγράφονται από την EmailQueue (π.χ. θα κρατήσουμε τα ίδια αρχεία). Εναλλακτικά, επειδή το email παραμένει διαθέσιμο στην ουρά, το ticket μπορεί να αναφέρεται πίσω στο EmailQueue entry για ανάκτηση συνημμένων on-demand.
* **Έλεγχοι Ακεραιότητας**: Ενσωματώνονται έλεγχοι για να αποφευχθεί διπλή εισαγωγή του ίδιου email: π.χ. constraint μοναδικότητας στο Ticket (με βάση κάποιο Message-ID του email ή συνδυασμό Sender+Subject+Date) ή έλεγχος αν ένα EmailQueue entry έχει ήδη status processed. Αυτό είναι σημαντικό ιδίως αν το σύστημα επανεκκινηθεί ενώ υπήρχαν μηνύματα στην ουρά.
* **Επεκτασιμότητα**: Η σχεδίαση της normalized δομής πρέπει να καλύπτει τις κοινές ανάγκες όλων των πελατών. Για πεδία που δεν είναι κοινού σχήματος, μπορεί να χρησιμοποιηθεί μια ευέλικτη προσέγγιση, όπως ένας πίνακας TicketProperties (key-value ανά ticket) ή JSON field για επιπλέον πληροφορίες.

### 5. Σήμανση Email ως Processed

Μετά την επιτυχημένη δημιουργία του ticket, το αρχικό email πρέπει να σημανθεί ως επεξεργασμένο ώστε να μην υποβληθεί ξανά. Αυτό γίνεται σε δύο επίπεδα:

* **Ενημέρωση Ουράς**: Στον πίνακα EmailQueue, το πεδίο Processed ενημερώνεται (π.χ. Processed = TRUE ή ProcessedDate = NOW() και ίσως TicketID για σύνδεση). Έτσι, η ουρά υποδεικνύει ότι αυτό το email ολοκλήρωσε τον κύκλο του.
* **Σήμανση στον Mail Server**: Για IMAP, η εφαρμογή μπορεί να μαρκάρει το μήνυμα ως Seen/Read ή να το μετακινήσει σε φάκελο π.χ. "Processed" ή "Archived". Το MailKit επιτρέπει τον χειρισμό των flags του μηνύματος ή τη μετακίνηση μεταξύ φακέλων (IMAP MOVE) μετά την επεξεργασία. Έτσι διασφαλίζεται ότι αν η υπηρεσία επανεκκινηθεί, δεν θα ξαναδιαβάσει τα ίδια emails από το Inbox. Για POP3, όπου δεν υπάρχουν flags, συνήθης πρακτική είναι η διαγραφή του email από τον server μετά τη λήψη εφόσον ολοκληρώθηκε η επεξεργασία (ή εναλλακτικά, διατήρηση τοπικής λίστας UIDLs των μηνυμάτων που έχουν ήδη ληφθεί).
* **Διαχείριση Σφαλμάτων**: Αν η επεξεργασία (κανόνες ή εισαγωγή) αποτύχει, το email μπορεί να **μην σημανθεί** ως processed ώστε να επανεπεξεργαστεί αφού διορθωθεί το σφάλμα. Σε τέτοια περίπτωση, ενδέχεται να μεταφερθεί προσωρινά σε φάκελο "Failed" ή να παραμείνει μη αναγνωσμένο. Επίσης, καταγράφεται σχετικό error log και (προαιρετικά) στέλνεται ειδοποίηση στον διαχειριστή.

### 6. Logging (Classic Mode)

* **Λεπτομερές Logging**: Σε κάθε στάδιο, το σύστημα πραγματοποιεί αναλυτική καταγραφή ενεργειών και αποτελεσμάτων:
  + Σύνδεση Mail: π.χ. "Connecting to IMAP server mail.example.com:993 (SSL)" και αποτέλεσμα (επιτυχία ή σφάλμα).
  + Λήψη Μηνυμάτων: π.χ. αριθμός μηνυμάτων που βρέθηκαν, UID κάθε μηνύματος που θα υποβληθεί σε επεξεργασία.
  + Αποθήκευση στην Ουρά: π.χ. "Email from X stored in queue with QueueID=123".
  + Εφαρμογή Κανόνων: Για κάθε πεδίο, μπορεί να καταγράφεται αν βρέθηκε τιμή ή όχι. Π.χ. "Rule 'OrderID regex' matched value 45678", "Rule 'Priority keyword' not found".
  + Εισαγωγή Ticket: π.χ. "Ticket created with TicketID=789 for QueueID=123".
  + Σήμανση email: π.χ. "Email QueueID=123 marked as processed and email moved to Processed folder".
* **Επίπεδα Καταγραφής**: Θα χρησιμοποιηθεί σύστημα ιεραρχημένου logging (Trace/Debug/Info/Warn/Error/Fatal). Π.χ. οι κανόνες μπορεί να είναι σε επίπεδο Debug (ώστε σε παραγωγικό σύστημα να μην γεμίζει το log με κάθε πεδίο), ενώ σφάλματα σύνδεσης/επεξεργασίας σε Error ή Warn.
* **Μορφή & Αποθήκευση Log**: Το logging διαμορφώνεται μέσω του κεντρικού configuration. Προβλέπεται καταγραφή σε αρχεία κειμένου (π.χ. rolling log files ημερήσια) και πιθανώς σε κεντρικό σύστημα (Event Log, Database ή ELK stack) ανάλογα με τις απαιτήσεις του πελάτη. Θα χρησιμοποιηθεί το built-in ILogger interface της .NET ή μια δημοφιλής βιβλιοθήκη όπως **Serilog** για δομημένο logging.
* **Διαγνωστικά & Παρακολούθηση**: Τα logs δίνουν την δυνατότητα στους developers/διαχειριστές να παρακολουθούν τη ροή της διαδικασίας και να διαγιγνώσκουν προβλήματα. Ιδιαίτερα σημαντική είναι η καταγραφή εξαιρέσεων (exceptions stack traces) σε περιπτώσεις σφαλμάτων, για γρήγορη διόρθωση bugs. Επίσης, key metrics όπως ο χρόνος επεξεργασίας κάθε email ή τυχόν ουρά αναμονής μπορούν να προκύπτουν από τα logs για αξιολόγηση απόδοσης.

### 7. Επεκτασιμότητα & Modular Υποδομή (Classic Mode)

* **Πολλαπλοί Λογαριασμοί & Νήματα**: Το Classic mode μπορεί να ρυθμιστεί να παρακολουθεί πολλαπλές θυρίδες email. Η υλοποίηση θα υποστηρίζει multi-threading ή async I/O, ώστε κάθε λογαριασμός να μπορεί να ελέγχεται ανεξάρτητα χωρίς καθυστέρηση. Π.χ. χρήση Task ανά λογαριασμό για παράλληλο έλεγχο ή ενός background service που εναλλάσσεται γρήγορα μεταξύ λογαριασμών.
* **Modular Connectors**: Αν και στο βασικό σενάριο, η έξοδος είναι ο normalized πίνακας tickets + κοινό API, το σύστημα σχεδιάζεται modular ώστε να μπορεί εύκολα να συνδεθεί σε διαφορετικές τελικές πλατφόρμες. Π.χ. θα μπορούσε να υπάρξει μια κλάση ITicketExporter interface με μεθόδους Export(ticket) και διαφορετικές υλοποιήσεις για διαφορετικά συστήματα (π.χ. εξαγωγή σε CRM A, σε Service Desk B, κλπ). Έτσι, η Classic mode θα μπορούσε εναλλακτικά να προωθεί απευθείας το ticket σε εξωτερικό API αντί για απλή εγγραφή σε DB, ανάλογα με το module που φορτώνεται.
* **Κλιμάκωση**: Προβλέπεται η δυνατότητα οριζόντιας κλιμάκωσης (horizontal scaling) μέσω πολλαπλών instances της υπηρεσίας, όπου κάθε instance μπορεί να αναλαμβάνει συγκεκριμένους λογαριασμούς email ή να μοιράζεται το φόρτο (χρησιμοποιώντας την ουρά DB ως κοινό σημείο αναφοράς). Χρήση locking/transactions στην DB θα εξασφαλίζει ότι δύο instances δεν επεξεργάζονται το ίδιο email.
* **Απόδοση**: Για αυξημένο όγκο email, το σύστημα πρέπει να βελτιστοποιηθεί: χρήση ενδεχομένως batch retrieval (IMAP permits batch fetch), χρήση indicies στη DB για τα κριτήρια αναζήτησης μη επεξεργασμένων μηνυμάτων, και καθαρισμός παλαιών εγγραφών ουράς ώστε η βάση να παραμένει αποδοτική.

## Λειτουργία AI Mode

Το AI Mode αξιοποιεί προηγμένα μοντέλα τεχνητής νοημοσύνης (GPT-4 με δυνατότητα Vision) για να διαβάσει και να κατανοήσει το περιεχόμενο των emails και των συνημμένων, εξάγοντας δυναμικά τα απαιτούμενα πεδία χωρίς ρητά καθορισμένους κανόνες. Η ροή μοιάζει με την Classic mode στα αρχικά και τελικά στάδια, με τη διαφορά ότι το στάδιο επεξεργασίας γίνεται μέσω κλήσης σε AI υπηρεσία.

### 1. Σύνδεση & Λήψη Emails (ίδιο με Classic)

* Το AI mode χρησιμοποιεί την **ίδια διαδικασία** σύνδεσης σε mail server (IMAP/POP3) και **αποθήκευσης στην Queue DB** όπως περιγράφηκε στο Classic mode. Δεν υπάρχει διαφοροποίηση στον τρόπο συλλογής των emails: και εδώ τηρείται η ουρά με σώμα email και συνημμένα.
* Συνεπώς, τα βήματα Classic Mode: Mail Server Connection, Queue Insertion, Marking processed (pending), εφαρμόζονται ομοιόμορφα. Η κοινή υλοποίηση εξασφαλίζει ότι ανεξαρτήτως mode, όλα τα εισερχόμενα θα φτάνουν στην EmailQueue με τον ίδιο τρόπο, έτοιμα προς επεξεργασία.

### 2. Επεξεργασία Περιεχομένου με AI (GPT-4 Vision)

* **Ενεργοποίηση AI**: Για κάθε email που βρίσκεται στην ουρά (και προορίζεται για AI επεξεργασία), το σύστημα θα συνθέτει ένα αίτημα προς την υπηρεσία GPT-4 (OpenAI API). Χρησιμοποιείται το GPT-4 model με δυνατότητα Vision, ώστε να μπορεί να επεξεργαστεί τόσο κείμενα όσο και εικόνες.
* **Συλλογή Δεδομένων για Ανάλυση**:
  + Κείμενο email: Το κείμενο/HTML σώμα του email καθαρίζεται (π.χ. αφαίρεση HTML tags αν χρειάζεται) και περιλαμβάνεται στο prompt προς το AI. Μπορεί να σταλεί αυτούσιο ή εντός ενός μεγάλου string με κατάλληλες οδηγίες.
  + Συνημμένα αρχεία: Όλα τα συνημμένα επεξεργάζονται ανάλογα με τον τύπο τους:
    - Για **PDF/DOCX/TXT**: Γίνεται προσπάθεια απόσπασης κειμένου. Αν το PDF ή Word είναι σε αναγνώσιμη μορφή (όχι σκαναρισμένο ως εικόνα), τότε χρησιμοποιείται βιβλιοθήκη να εξαχθεί το αλφαριθμητικό περιεχόμενο. Αυτό το κείμενο προστίθεται στο prompt (πιθανώς με επισήμανση από ποιο αρχείο προήλθε).
    - Αν το PDF είναι σκαναρισμένη εικόνα ή αν έχουμε **εικόνες JPG/PNG**: Τα αρχεία αυτά αποστέλλονται ως εικόνες στο AI μοντέλο. Το GPT-4 Vision μπορεί να λάβει εικόνες εισόδου και να τις αναλύσει (π.χ. να εκτελέσει OCR ή περιγραφική κατανόηση). Για την αποστολή πολλαπλών εικόνων, το σύστημα μπορεί να σπάσει ένα πολυσέλιδο PDF σε μεμονωμένες εικόνες σελίδων ή να στείλει πολλαπλά image attachments αν υποστηρίζεται.
    - Η OpenAI API (2024+) επιτρέπει την μεταφόρτωση αρχείων ή την παροχή τους ως binary/image input μαζί με το prompt. Εναλλακτικά, σε περίπτωση περιορισμών, θα μπορούσε να γίνει χρήση εξωτερικού OCR και παροχή αποτελέσματος κειμένου.
* **Διαμόρφωση Prompt**: Το prompt προς το GPT-4 κατασκευάζεται δυναμικά και παραμετροποιείται ανά πελάτη:
  + Ξεκινά με μια **οδηγία** που εξηγεί στο μοντέλο το ρόλο του, π.χ.: "Act as an email ticketing assistant. You will read an email and its attachments and extract the following fields...".
  + Στη συνέχεια περιλαμβάνει το **κείμενο του email** (subject, body) και περίληψη/εξαγωγή κειμένου συνημμένων. Κάθε τμήμα μπορεί να προσημειώνεται, π.χ. "[EMAIL BODY] ... [ATTACHMENT1: PDF Text] ... [ATTACHMENT2: Image]".
  + Έπειτα, ακολουθούν **οδηγίες εξαγωγής πεδίων**. Αυτές προσαρμόζονται ανά πελάτη/περίπτωση. Παραδείγματα:
    - "Extract the customer's full name, contract number, and issue description from the above email."
    - "Provide the output in JSON with keys: "Title", "Description", "Priority"."
  + Δίνεται έμφαση ώστε το μοντέλο να **απαντήσει σε δομημένη μορφή** (JSON ή key-value pairs) για εύκολη αναλυτική εισαγωγή. (Η επιλογή JSON είναι ιδανική ώστε να γίνει parse αυτόματα η απάντηση.)
* **Κλήση API**: Η εφαρμογή καλεί το OpenAI API endpoint (π.χ. /v1/chat/completions για GPT-4) με κατάλληλο payload:
  + Model: "gpt-4" (ή variant που υποστηρίζει vision).
  + Messages: Το prompt ως user message (ή system+user messages αν χρησιμοποιηθεί η chat format).
  + Attachments: εφόσον το API το υποστηρίζει, περιλαμβάνονται ως μέρος του αιτήματος (π.χ. OpenAI έχει μηχανισμό files ή inline images).
  + Παρατηρήσεις: Θα χρησιμοποιηθεί η επίσημη βιβλιοθήκη OpenAI για .NET για ευκολία ενσωμάτωσης, η οποία υποστηρίζει όλες τις λειτουργίες του API συμπεριλαμβανομένου του GPT-4 και async/await κλήσεων.
* **Λήψη & Κατανόηση Απάντησης**:
  + Το AI θα επιστρέψει μια απάντηση που (ιδανικά) περιέχει τα ζητούμενα πεδία. Η εφαρμογή θα πρέπει να **επαληθεύσει** το format (π.χ. έγκυρο JSON) και να το μετατρέψει σε αντικείμενο/struct για περαιτέρω χρήση.
  + Σε περίπτωση που το AI δεν δώσει τα δεδομένα σε σωστή μορφή (π.χ. απάντησε με ελεύθερο κείμενο ή παρέλειψε κάποια πεδία), το σύστημα μπορεί να επιχειρήσει **διόρθωση**: ενδεχομένως νέα κλήση με αυστηρότερη οδηγία format ή χρήση post-processing (π.χ. regular expression πάνω στην απάντηση του AI για άντληση τιμών).
  + Σημαντικό: Δεδομένου ότι τα μοντέλα μπορεί να παράγουν και λάθη, μπορεί να χρειαστεί validation ορισμένων πεδίων. Π.χ. αν αναμένεται αριθμός ticket να είναι ακέραιος, γίνεται δοκιμή parse. Αν αποτύχει, μπορούμε να καταγράψουμε προειδοποίηση ότι η AI εξαγωγή ίσως δεν είναι αξιόπιστη για το συγκεκριμένο πεδίο.
* **Χρόνος & Κόστη**: Η κλήση στο GPT-4 είναι σχετικά αργή (δεκάδες έως εκατοντάδες ms, ανάλογα με το μέγεθος του κειμένου και εικόνων) και έχει κόστος. Η υλοποίηση θα πρέπει:
  + Να είναι **ασύγχρονη** (async) ώστε να μην μπλοκάρει το υπόλοιπο σύστημα. Ίσως χρησιμοποιηθεί ουρά εργασιών και περιορισμός στο πόσα παράλληλα API calls γίνονται (λόγω rate limits).
  + Να υποστηρίζει πιθανή **επανάληψη (retry)** με exponential backoff αν υπάρξουν προσωρινές αποτυχίες API (π.χ. timeout ή rate limit εξάντληση).
  + Να λαμβάνει υπόψη τυχόν μηνύματα σφάλματος από το AI (π.χ. αν σταλεί πολύ μεγάλο prompt και απορριφθεί για token limit, ίσως χρειαστεί να κόψει μέρος του κειμένου ή να αγνοήσει κάποια συνημμένα και να ξαναδοκιμάσει).

### 3. Εξαγωγή Πεδίων & Καταχώριση Ticket (AI Mode)

* **Χαρτογράφηση Πεδίων**: Όταν ληφθεί η απάντηση του AI, το σύστημα θα χαρτογραφήσει τα εξαγμένα πεδία στα πεδία του normalized πίνακα Tickets. Στην ιδανική περίπτωση που χρησιμοποιείται JSON output, γίνεται map JSON keys -> στήλες πίνακα.
* **Παραμετροποίηση Ανά Πελάτη**: Όπως και στους κανόνες, έτσι και εδώ κάθε πελάτης έχει διαφορετικές απαιτήσεις. Το prompt ήδη φροντίζει να ζητήσει συγκεκριμένα πεδία. Επιπλέον, μπορεί να υπάρχει ένα configuration που ορίζει:
  + Ποια keys να αναμένουμε από το AI για τον συγκεκριμένο πελάτη.
  + Default τιμές ή μετασχηματισμοί (π.χ. αν το AI δίνει priority ως "High/Low", να μετατραπεί σε αριθμητική κλίμακα 1-5).
  + Ενέργειες αν λείπει κάποιο πεδίο (π.χ. το πεδίο "ContractNumber" δεν βρέθηκε -> θέσε το ως null και συνέχισε ή σήμανε σφάλμα).
* **Εισαγωγή στο Tickets DB**: Η εισαγωγή μοιάζει με του Classic mode:
  + Δημιουργείται νέα εγγραφή στον πίνακα Tickets με όλα τα πεδία που βρέθηκαν από το AI.
  + Προστίθεται συσχέτιση συνημμένων (συνήθως αντιγραφή των συνημμένων από το EmailQueue στην Ticket εγγραφή, παρότι το AI τα έχει ήδη "δει", τα αρχεία πρέπει να είναι διαθέσιμα και στο ticket για τον τελικό χρήστη).
  + Καταγράφεται το **ClientID/System** ώστε το API να μπορεί να φιλτράρει tickets ανά τελικό σύστημα.
* **Επιβεβαίωση & Μετα-επεξεργασία**: Μετά την εισαγωγή, το σύστημα μπορεί να εκτελεί κάποιους αυτοματοποιημένους ελέγχους ποιότητας:
  + Π.χ. αν το AI επέστρεψε τιμή για πεδίο που δεν αναμενόταν (κλειδί εκτός configuration), να αγνοηθεί ή να μπει σε κάποιο πεδίο extra.
  + Έλεγχος ότι κρίσιμα πεδία (π.χ. Title, Description) δεν είναι κενά. Αν είναι, ίσως δημιουργηθεί ticket με ένα γενικό τίτλο "Unknown" και σημείωση ότι απαιτείται ανθρώπινη παρέμβαση.
  + Εφόσον όλα είναι εντάξει, προχωρά η σήμανση processed.

### 4. Σήμανση Email ως Processed (AI Mode)

* Η διαδικασία εδώ είναι **όμοια με το Classic mode**. Το email που μόλις επεξεργάστηκε από το AI:
  + Σημαίνεται στην EmailQueue ως processed (status ενημέρωση και ενδεχομένως αποθήκευση του TicketID που δημιουργήθηκε).
  + Στον mail server, μαρκάρεται ως read/seen ή μεταφέρεται σε φάκελο αρχείου, ή διαγράφεται αν αυτό έχει καθοριστεί, ώστε να μην ξαναδιαβαστεί.
* Σε περίπτωση σφάλματος (π.χ. το AI δεν έδωσε αποτέλεσμα ή απέτυχε η κλήση):
  + Μπορεί να εφαρμοστεί fallback: είτε να μπει σε επανάληψη (retry later), είτε να μεταπέσει στο Classic mode (αν υπάρχουν διαθέσιμοι κανόνες ως εφεδρεία), είτε να μείνει στην ουρά ως μη επεξεργασμένο με flagged status για έλεγχο.
  + Τα μη επεξεργασμένα emails πρέπει να είναι ανιχνεύσιμα ώστε να μην χαθούν (π.χ. μια ξεχωριστή λίστα "errors").

### 5. Logging (AI Mode)

Το Logging στο AI mode επεκτείνεται ώστε να καλύψει ειδικά και τις αλληλεπιδράσεις με το AI:

* **API Request/Response Logs**: Κάθε κλήση προς το GPT API θα καταγράφεται:
  + Χρόνος κλήσης, διάρκεια μέχρι απόκριση.
  + Μέγεθος prompt (πόσοι χαρακτήρες ή tokens εστάλησαν) και ενδεχομένως τα IDs των attachments που στάλθηκαν.
  + Status απόκρισης (επιτυχία ή τύπος σφάλματος).
* **Καταγραφή Prompt & Response**: Για λόγους διαγνωστικούς, μπορεί να καταγραφεί το ίδιο το prompt που στάλθηκε και η απάντηση που λήφθηκε. **Προσοχή:** Επειδή αυτά μπορεί να περιέχουν ευαίσθητα δεδομένα (περιεχόμενο email), πρέπει να προστατεύονται. Ίσως αποθηκεύονται σε ασφαλή τοποθεσία με πρόσβαση μόνο από admins, ή σε πίνακα AI\_Logs στη DB. Αυτό είναι πολύτιμο για ανασκόπηση της ποιότητας του AI (π.χ. όταν ένα ticket βγει λάθος, να δει ο developer τι ζήτησε και τι κατάλαβε το μοντέλο).
* **Συγκριτικό Logging**: Μπορεί να καταγράφεται σύγκριση μεταξύ Classic και AI αν υπάρχει (π.χ. αν παράλληλα τρέχουν για δοκιμή). Στη φάση παραγωγής όμως συνήθως θα τρέχει μόνο ένα mode.
* **Άλλα**: Όλα τα βήματα όπως και πριν (λήψη email, εισαγωγή DB κλπ) επίσης καταγράφονται. Οι διαφορές είναι στα επιπλέον logs της AI επικοινωνίας. Παραδείγματα:
  + "AI Mode: Sending email QueueID=125 content to GPT-4".
  + "AI Mode: Received response for QueueID=125 (tokens used=500, cost=$0.03)" – εφόσον παρακολουθούνται και τα κόστη.
  + "AI Mode: Extracted fields: Title='Issue with Order', Priority='High', ...".
* **Ρυθμίσεις Logging**: Ομοίως υπάρχει δυνατότητα ρύθμισης επιπέδων. Σε περιβάλλον παραγωγής ίσως επιλέξουμε να **μην** καταγράφεται πλήρες περιεχόμενο prompt/response παρά μόνο περιλήψεις ή key results, για λόγους ασφάλειας και όγκου. Σε test mode θα καταγράφονται λεπτομερώς για βελτίωση των prompts.

### 6. Κοινή Υποδομή & Εναλλαγή Λειτουργίας

* Το AI mode μοιράζεται σε μεγάλο βαθμό την υποδομή με το Classic mode. Η υλοποίηση θα χρησιμοποιήσει το **Strategy Pattern** ή παρόμοιο, ώστε να υπάρχει ένα interface π.χ. IEmailProcessor με μέθοδο ProcessEmail(queueItem) το οποίο υλοποιείται από δύο κλάσεις: ClassicProcessor και AIProcessor. Το ποια χρησιμοποιείται καθορίζεται από το configuration ή context του κάθε πελάτη.
* **Εναλλαγή Mode**: Σε επίπεδο εφαρμογής, μπορεί να υπάρχει global ρύθμιση (π.χ. "ProcessingMode": "AI" ή "Classic") στο κεντρικό config που καθορίζει την προεπιλεγμένη λειτουργία. Επιπλέον, θα μπορούσε να υποστηριχθεί μίξη – π.χ. ορισμένοι πελάτες να χρησιμοποιούν AI και άλλοι classic κανόνες. Σε αυτή την περίπτωση, η επιλογή mode θα ήταν property στο προφίλ/παραμετροποίηση κάθε πελάτη.
* **Κοινό Configuration/Logging**: Η υπηρεσία χρησιμοποιεί τις ίδιες ρουτίνες για ανάκτηση email, εισαγωγή στην ουρά, database access και logging. Αυτό μειώνει την πιθανότητα διπλού κώδικα και σφαλμάτων. Οι διαφορές εντοπίζονται μόνο στο κομμάτι "πώς από το email βγάζουμε τα ticket fields".
* **Επεκτασιμότητα**: Η modular προσέγγιση σημαίνει ότι στο μέλλον μπορεί να προστεθεί και τρίτος τρόπος επεξεργασίας (π.χ. **Hybrid Mode** ή με άλλου είδους AI μοντέλο) χωρίς να ξαναγραφτεί η υπόλοιπη εφαρμογή. Αρκεί η προσθήκη νέας κλάσης που υλοποιεί το interface επεξεργασίας και η κατάλληλη ρύθμιση στο config.

## Κεντρική Παραμετροποίηση Συστήματος

Όλες οι ρυθμίσεις του συστήματος διαχειρίζονται μέσω κεντρικού αρχείου ρυθμίσεων (π.χ. appsettings.json αν πρόκειται για .NET Core app) ή/και σχετικής εγγραφής στη βάση:

* **Ρυθμίσεις Mail Server**: Λίστα από account configurations, π.χ.:

|  |
| --- |
| "MailAccounts": [  {  "ClientID": "ABC\_Corp",  "Protocol": "IMAP",  "Server": "imap.mailserver.com",  "Port": 993,  "UseSSL": true,  "Username": "support@abccorp.com",  "Password": "\*\*\*\*\*\*",  "PollIntervalMinutes": 5,  "Mode": "AI"  },  { ... δεύτερος λογαριασμός ... }  ] |

 Κάθε λογαριασμός μπορεί να έχει επιπλέον ρυθμίσεις όπως φάκελο προορισμού (Inbox ή άλλο) και ίσως field mapping config reference.

 **Ρυθμίσεις Βάσης Δεδομένων**: Στοιχεία σύνδεσης (connection strings) για την Queue DB και Ticket DB (μπορεί να είναι η ίδια ή διαφορετική). Επειδή υποστηρίζονται διαφορετικά RDBMS (MS SQL, MySQL, PostgreSQL), μπορεί να χρειάζεται provider name ή ξεχωριστά connection configs. Ιδανικά θα χρησιμοποιηθεί ORM όπως Entity Framework ή Dapper που αφαιρεί τις λεπτομέρειες.

 **Mode & AI Settings**: Global διακόπτες για εναλλαγή mode (αν όχι ορισμένο ανά account). Ρυθμίσεις AI όπως:

* OpenAI API Key (securely stored).
* Model version (π.χ. gpt-4 vs gpt-4-32k αν υπάρχει επιλογή).
* Max tokens, temperature, κ.λπ. παράμετροι του μοντέλου.
* Timeout για API calls.

 **Extraction Rules Config**: Σε Classic mode, οι κανόνες ανά πελάτη μπορούν να οριστούν σε εξωτερικά αρχεία (π.χ. JSON/YAML) ή πίνακες DB. Ενδεικτικά, ένα JSON schema για κανόνες:

|  |
| --- |
| "ExtractionRules": {  "ABC\_Corp": [  { "Field": "Title", "Source": "Subject", "Pattern": null },  { "Field": "CustomerName", "Source": "Body", "Pattern": "Customer:\\s\*(.+)" },  { "Field": "Contract", "Source": "Attachment:contract.pdf", "Pattern": "Contract No:\\s\*(\\w+)" }  ],  "XYZ\_Ltd": [ ... άλλο σύνολο κανόνων ... ]  } |

* Το σύστημα θα φορτώνει αυτούς τους κανόνες κατά την εκκίνηση ή από cache.
* **AI Prompt Templates**: Παρόμοια, για AI mode, μπορεί να υπάρχουν templates ή οδηγίες ανά πελάτη:
  + Στα configuration θα αποθηκεύεται το βασικό prompt ανά πελάτη (ή και λίστα δειγμάτων Q&A για few-shot αν χρειάζεται).
  + Ποια πεδία ζητάμε από το AI ανά πελάτη (για επιβεβαίωση ότι ταιριάζουν με τα Ticket fields).
  + Γλώσσα επικοινωνίας με AI (αν τα emails είναι π.χ. στα ελληνικά, το prompt ίσως πρέπει να είναι δίγλωσσο ή να υποδεικνύει στο μοντέλο ότι μπορεί να απαντήσει στα αγγλικά για σταθερότητα κλπ).
* **Logging & Debug Options**: Επίπεδο logging (π.χ. Info, Debug), μέγιστο μέγεθος log αρχείων, κλπ. Επίσης, επιλογές debug mode που ίσως ενεργοποιούν επιπλέον ελέγχους ή καταγραφές (π.χ. πλήρες AI response dumping).
* **API Settings**: Ρυθμίσεις για τον web server/API (πόρτα, authentication keys, CORS κ.λπ., δες παρακάτω).

Το κεντρικό αρχείο παραμετροποίησης είναι δομημένο ώστε να μπορεί εύκολα να επεκταθεί με νέες ρυθμίσεις. Οι αλλαγές σε αυτό (όπως προσθήκη νέου mail account ή αλλαγή κανόνων) θα πρέπει να εφαρμόζονται είτε με restart της υπηρεσίας είτε (ιδανικότερα) με dynamic reload χωρίς διακοπή (hot reload των config, π.χ. χρήση IOptionsSnapshot σε .NET).

## API Endpoint για Τελικά Συστήματα Ticketing

Για να ενσωματωθούν τα παραγόμενα tickets σε εξωτερικά συστήματα (π.χ. CRM, ITSM πλατφόρμες, helpdesk software), το σύστημα "Email to Ticket" παρέχει ένα **RESTful API endpoint** από όπου μπορούν να ανακτηθούν τα νέα ή ενημερωμένα tickets.

### 1. Σκοπός & Λειτουργικότητα API

* Επιτρέπει στα τελικά συστήματα να κάνουν **polling** ή ερώτηση περιοδικά για νέα tickets που τα αφορούν, αντί αυτά να έχουν άμεση πρόσβαση στη βάση δεδομένων.
* Διατηρεί το **decoupling** μεταξύ της υπηρεσίας email processing και των πολλαπλών καταναλωτών των δεδομένων.
* Μπορεί επίσης να χρησιμοποιηθεί για ενοποίηση με webhooks: π.χ. το σύστημα θα μπορούσε με ένα μικρό customization να καλεί το API του εκάστοτε ticketing συστήματος όταν δημιουργείται νέο ticket, αλλά η out-of-the-box λύση είναι τα εξωτερικά συστήματα να ρωτούν αυτό το API.

### 2. Διαθέσιμα Endpoint & Λειτουργίες

* **GET /api/tickets**: Επιστρέφει λίστα tickets. Υποστηρίζει παραμέτρους φιλτραρίσματος:
  + since\_id ή since\_datetime για να λαμβάνει μόνο tickets νεότερα από κάποιο ID ή ημερομηνία (έτσι υλοποιείται incremental polling).
  + client\_id ή άλλο φίλτρο ώστε ένα σύστημα να τραβά μόνο τα δικά του σχετικά tickets (π.χ. το CRM A να παίρνει μόνο όσα ClientID=CRM\_A).
  + Πιθανές άλλες παράμετροι: status (new/processed), περιορισμός πλήθους (batch size, pagination).
* **GET /api/tickets/{id}**: Επιμέρους ανάκτηση συγκεκριμένου ticket, συμπεριλαμβανομένων λεπτομερειών και συνημμένων.
* **POST /api/tickets/acknowledge** (ή παρόμοιο): Θα μπορούσε να υπάρχει ένα endpoint όπου ένα τελικό σύστημα δηλώνει ότι παρέλαβε συγκεκριμένα tickets, ώστε το Email2Ticket σύστημα να τα μαρκάρει ως παραδοθέντα. Αυτό είναι χρήσιμο αν θέλουμε να αποφεύγουμε την εκ νέου αποστολή των ίδιων (ειδικά αν δεν χρησιμοποιούμε τη μέθοδο since\_id).
* **Auth**: Η πρόσβαση στο API θα προστατεύεται είτε με API keys ανά πελάτη/σύστημα είτε με OAuth2. Π.χ. κάθε client integration έχει ένα token που πρέπει να στείλει (HTTP header) για να αυθεντικοποιηθεί. Οι κλήσεις θα γίνονται πάντα μέσω HTTPS για ασφάλεια.
* **Response Format**: Τα δεδομένα επιστρέφονται σε JSON. Παράδειγμα απάντησης για GET /api/tickets?client\_id=ABC\_Corp&since\_id=1000:

|  |
| --- |
| [  {  "TicketID": 1005,  "ClientID": "ABC\_Corp",  "Title": "Order Delay Issue",  "Description": "Customer reports a delay in order #1234...",  "Priority": "High",  "CreatedDate": "2025-05-13T07:45:00Z",  "Attachments": [  { "FileName": "screenshot.png", "Url": "https://<host>/api/tickets/1005/attachments/1" }  ]  },  { ... άλλο ticket ... }  ] |

* Τα attachments μπορούν είτε να ενσωματωθούν ως Base64 (λιγότερο επιθυμητό λόγω όγκου) είτε ως URLs σε ξεχωριστό endpoint (GET /api/tickets/{id}/attachments/{attId}) όπου θα γίνεται download.
* **Modular Support**: Αν απαιτείται διαφορετικό interface (π.χ. SOAP web service ή GraphQL) για κάποια ολοκλήρωση, η αρχιτεκτονική modular του Integration layer επιτρέπει την προσθήκη επιπλέον παρουσιάσεων. Ωστόσο, η κύρια παρεχόμενη μορφή θα είναι REST/JSON ως η πιο ευρέως συμβατή.

### 3. Ενέργειες μετά την Ανάκτηση

* Όταν ένα εξωτερικό σύστημα τραβάει επιτυχώς ένα ticket μέσω του API, δύο προσεγγίσεις είναι δυνατές:
  + **Pull-only μοντέλο**: Το Email2Ticket δεν αλλάζει τίποτα στο ticket (μένει ως έχει). Το εξωτερικό σύστημα μπορεί να καλεί περιοδικά για τυχόν νέα tickets. Εδώ το since\_id ή since\_date παίζει ρόλο για να μην επαναφέρει παλιά.
  + **Acknowledge μοντέλο**: Μετά την ανάκτηση, το εξωτερικό σύστημα καλεί π.χ. acknowledge endpoint ή στέλνει πίσω ένα flag ώστε το Email2Ticket να μαρκάρει το ticket ως "Delivered". Έτσι, σε επόμενες κλήσεις δεν θα εμφανίζεται. Αυτό αποτρέπει διπλή καταχώριση στο εξωτερικό σύστημα σε περίπτωση που για κάποιο λόγο γίνουν επαναλαμβανόμενες κλήσεις χωρίς σωστό since\_id.
* Η επιλογή μοντέλου μπορεί να καθορίζεται από το integration. Αν το εξωτερικό σύστημα έχει δυνατότητα να θυμάται το τελευταίο ID που έλαβε, το pull-only με since\_id είναι απλούστερο. Αν όχι, ίσως χρειάζεται το acknowledge για σιγουριά.

### 4. Rate Limiting & Performance

* Το API θα σχεδιαστεί ώστε να **αντέχει σε συχνές κλήσεις** (π.χ. ένα helpdesk μπορεί να ρωτάει κάθε λεπτό). Θα χρησιμοποιηθεί caching όπου εφικτό και βελτιστοποίηση ερωτημάτων στη DB. Π.χ. index στο πεδίο CreatedDate για γρήγορο εύρεση νέων records.
* Επίσης δύναται να εφαρμοστεί **rate limiting** ανα πελάτη API key για αποφυγή κατάχρησης, με κατάλληλα μηνύματα σφάλματος (429 Too Many Requests).
* Η **κωδικοποίηση δεδομένων** (JSON) θα είναι σε UTF-8. Προσοχή σε ειδικούς χαρακτήρες από τα emails (θα αποθηκεύονται σωστά στη DB με UTF8 και θα επιστρέφονται ορθά).
* Αν μεταφέρονται μεγάλοι όγκοι δεδομένων (π.χ. πολλά attachments), μπορεί να ενθαρρυνθεί η χρήση συμπίεσης HTTP (Accept-Encoding: gzip).

## Απαιτήσεις Λογισμικού & Εργαλεία Υλοποίησης

* **Γλώσσα & Πλαίσιο**: Η εφαρμογή θα υλοποιηθεί σε C# (.NET 8 ή μεταγενέστερο για cross-platform υποστήριξη και βελτιώσεις απόδοσης). Θα είναι κατά προτίμηση μια Windows Service (ή Linux daemon) χωρίς UI, που τρέχει συνεχώς στο υπόβαθρο. Εναλλακτικά μπορεί να υλοποιηθεί ως ASP.NET Core Hosted Service σε συνδυασμό με το API.
* **Βιβλιοθήκες Email**: Θα χρησιμοποιηθεί το **MailKit/MimeKit** library για τη διαχείριση IMAP/POP3 και parsing των emails, καθώς παρέχει ώριμη υλοποίηση των πρωτοκόλλων και υποστήριξη SSL/TLS, OAuth2 κλπ. Το MailKit θα αξιοποιηθεί και για την εύκολη ανάγνωση σώματος/συνημμένων (MimeKit part handling).
* **OpenAI API**: Για την ενσωμάτωση με GPT-4, θα χρησιμοποιηθεί η **επίσημη βιβλιοθήκη OpenAI .NET** (OpenAI SDK), που καλύπτει τα endpoints του OpenAI και υποστηρίζει ασύγχρονες κλήσεις. Σε περίπτωση που απαιτείται λειτουργία vision (εικόνες), θα εξεταστεί είτε η δυνατότητα της βιβλιοθήκης για image upload είτε θα χρησιμοποιηθεί εναλλακτική προσέγγιση (π.χ. απευθείας REST κλήσεις με HttpClient για multipart form data).
* **Βάση Δεδομένων**: Το σύστημα θα είναι αγνωστικό βάσης μέσω ORM ή data access layer. Προτείνεται **Entity Framework Core** για ευκολία συντήρησης, με migration scripts για δημιουργία των πινάκων σε MS SQL, MySQL ή PostgreSQL. Εναλλακτικά, για απόλυτο έλεγχο και ταχύτητα, μπορεί να χρησιμοποιηθεί **Dapper** ή καθαρές SQL εντολές μέσω DbConnection. Θα δοθεί σημασία στη σωστή χαρτογράφηση τύπων δεδομένων για συμβατότητα (π.χ. χρήση NVARCHAR για Unicode κείμενο, VARBINARY για BLOBs).
* **Logging Framework**: Χρήση του built-in **Microsoft.Extensions.Logging** σε συνδυασμό με provider π.χ. **Serilog** για καταγραφή σε αρχείο και κονσόλα. Το logging subsystem θα ρυθμίζεται από το appsettings (όπως προαναφέρθηκε) και θα επιτρέπει εύκολη αλλαγή format (JSON logs vs plaintext).
* **JSON Handling**: Για την ανάλυση/δημιουργία JSON (π.χ. στα API responses, στο parsing των AI outputs), θα χρησιμοποιηθεί είτε το ενσωματωμένο System.Text.Json είτε η δημοφιλής βιβλιοθήκη **Newtonsoft.Json** αν χρειαστούν επιπλέον δυνατότητες.
* **Testing**: Προβλέπεται η υλοποίηση μονάδων ελέγχου (unit tests) ειδικά για:
  + Τη λειτουργία των extraction rules (Classic): δοκιμή regex σε δείγματα email.
  + Τη σωστή δημιουργία prompt και parsing απάντησης (AI): πιθανώς με mock του OpenAI API.
  + Το API endpoint: δοκιμή ότι φέρνει σωστά δεδομένα φιλτραρισμένα.
* **Περιβάλλον Εκτέλεσης**: Η εφαρμογή πρέπει να μπορεί να τρέξει ως υπηρεσία. Σε Windows, θα εγκαθίσταται ως Windows Service (π.χ. μέσω Topshelf ή PowerShell installation). Σε Linux/docker, θα τρέχει συνεχώς ως console app με επίβλεψη (systemd ή container orchestration).
* **Ασφάλεια**: Ιδιαίτερη προσοχή θα δοθεί στην ασφάλεια δεδομένων:
  + Τα διαπιστευτήρια email και API keys δεν θα είναι σε απλό κείμενο στο config. Θα διερευνηθεί η χρήση ασφαλών stores (π.χ. **Key Vault** ή κρυπτογράφηση του config section).
  + Τα ευαίσθητα δεδομένα email (περιεχόμενο, συνημμένα) προστατεύονται στον αποθηκευτικό χώρο και στο log (αποφυγή εκτύπωσης περιεχομένου σε αρχεία log σε παραγωγή).
  + Συμμόρφωση με GDPR/ιδιωτικότητα: διαγραφή προσωπικών δεδομένων κατόπιν αιτήματος, ενδεχομένως ρυθμιζόμενη χρονική διατήρηση των email στη βάση (π.χ. καθαρισμός EmailQueue entries μετά από Χ μήνες).
* **Σφάλματα & Ανάκαμψη**: Εκτός από το logging, θα υλοποιηθούν μηχανισμοί αντοχής:
  + **Retry policies** για συνδέσεις email (π.χ. αν ο mail server δεν ανταποκρίνεται, επανάληψη μετά από 30 δευτερόλεπτα).
  + **Circuit breaker** για το AI API: αν αποτυγχάνουν συνεχόμενες κλήσεις, παύση προσπαθειών για λίγο για να μην επιβαρύνεται περαιτέρω και fallback σε alternative (ίσως Classic) εάν δυνατό.
  + **Dead-letter queue** concept: αν ένα συγκεκριμένο email αποτυγχάνει να επεξεργαστεί μετά από π.χ. 3 προσπάθειες, να μεταφέρεται σε έναν πίνακα "FailedEmails" με λόγο αποτυχίας.
  + **Monitoring & Alerts**: Η εφαρμογή θα εκθέτει βασικά στατιστικά (ίσως μέσω του ίδιου API ή logs) ώστε ένα εξωτερικό σύστημα παρακολούθησης να ειδοποιεί αν, π.χ., δεν έχουν εισαχθεί νέα tickets για Χ ώρες (πιθανή δυσλειτουργία), ή αν παρουσιάζονται επαναλαμβανόμενα errors.

## Συμπέρασμα

Το σύστημα "Email to Ticket" συνδυάζει την αξιοπιστία των κλασικών μεθόδων ανάλυσης emails (Classic Mode) με την ευελιξία της τεχνητής νοημοσύνης (AI Mode) για την κάλυψη ποικίλων σεναρίων πελατών. Με την ανωτέρω αρχιτεκτονική και προδιαγραφές, διασφαλίζεται ότι:

* Θα υποστηρίζει ασφαλή και αποτελεσματική ανάγνωση emails (POP3/IMAP με SSL/TLS).
* Θα αποθηκεύει όλα τα απαραίτητα δεδομένα (σώμα, συνημμένα) δομημένα σε ουρά για αξιόπιστη επεξεργασία.
* Θα εφαρμόζει κανόνες εξαγωγής ή AI ανάλυση αναλόγως, προσφέροντας **παραμετροποίηση ανά πελάτη** και ακρίβεια στην μετατροπή email σε ticket.
* Θα διατηρεί ολοκληρωμένο ιστορικό και διαφάνεια μέσω εκτενούς logging, τόσο για debugging όσο και για auditing (ειδικά στις AI αποφάσεις).
* Θα ενσωματώνεται εύκολα με άλλα συστήματα μέσω ενός ανοιχτού API, καθιστώντας το μια ανεξάρτητη αλλά συνεργάσιμη υπηρεσία.
* Θα είναι επεκτάσιμο, δίνοντας τη δυνατότητα στους C# developers να προσθέσουν νέες δυνατότητες, κανόνες ή μοντέλα AI, και να υποστηρίξουν περισσότερα συστήματα ή μεγαλύτερο φορτίο email στο μέλλον.

Με αυτές τις προδιαγραφές, οι developers μπορούν να προχωρήσουν στην υλοποίηση γνωρίζοντας τις απαιτήσεις και τα σημεία προσοχής, επιτυγχάνοντας ένα ισχυρό σύστημα μετατροπής Email σε Ticket για επιχειρησιακή χρήση.

