Πάγκος Εργασίας Εκπομπών Mærsk

Γκίνης Κωνσταντίνος

12 Φεβρουαρίου 2025

Περιεχόμενα

1 Η ανάγκη δημιουργίας του Πάγκου Εργασίας Εκπομπών **3**
- 1.1 Μέτρηση εκπομπών CO_2 . . . 3
- 1.2 Η προηγούμενη διαδικασία και τα μειονεκτήματά της . . . 5
- 1.3 Το λογισμικό μιας ολοκληρωμένης λύσης . . . 5
- 1.4 Περιορισμοί . . . 6

2 Τεχνολογία **7**
- 2.1 Επιλογή τεχνολογιών . . . 7
  - 2.1.1 Πλεονεκτήματα BEAM - Elixir . . . 8
- 2.2 Αρχιτεκτονική . . . 9

3 Εργασιακές μέθοδοι **11**
- 3.1 Ανάπτυξη λογισμικού με μεθόδους Agile - Extreme Programming (XP) . . . 11
  - 3.1.1 Pair Programming . . . 11
  - 3.1.2 TDD - Test Driven Development . . . 12
  - 3.1.3 CI / CD - Continuous Integration / Continuous Delivery . . . 13
  - 3.1.4 Vertical Ownership (κατακόρυφη ιδιοκτησία) . . . 13
  - 3.1.5 Feedback loops (βρόγχοι ανατροφοδότησης) . . . 14

4 Βιβλιογραφία **16**

Ο Πάγκος Εργασίας Εκπομπών (Emissions Workbench) είναι ένα σύστημα λογισμικού της Mærsk με στόχο τη διευκόλυνση της επίτευξης του στόχου της εταιρείας για καθαρές μηδενικές εκπομπές το 2040 (net zero 2040). Ξεκίνησε την ανάπτυξή του το Μάρτιο του 2023 και απέκτησε τους πρώτους χρήστες το Σεπτέμβριο του ίδιου έτους.

Κεφάλαιο 1

Η ανάγκη δημιουργίας του Πάγκου Εργασίας Εκπομπών

Αναμένεται σύντομα, κάθε εταιρεία εντός Ευρωπαϊκής Ένωσης να υποχρεωθεί να δηλώνει τις εκπομπές διοξειδίου του άνθρακα (CO_2 από δω και πέρα) που προκύπτουν απ' τη διεξαγωγή των επιχειρηματικών ενεργειών της.

Συγκεκριμένα η διαχειρίστρια εταιρεία της Mærsk (holding group), έχει δείξει έμπρακτα με μεγάλο μέρος επενδύσεων, πως έχει ως προτεραιότητα τη μείωση της επιρροής στο περιβάλλον από τις επιχηριματικές της δραστηριότητες. Κατ' επέκταση έχει θέσει ως στόχο την επίτευξη καθαρών μηδενικών εκπομπών μέχρι το 2040 (αυτό συνεπάγεται ότι ο κύκλος του άνθρακα που συσχετίζεται με τις δραστηριότητες της εταιρείας, θα έχει μηδενικό ισοζύγιο, δηλαδή δεν θα προστίθεται CO_2 στην ατμόσφαιρα).

Για την επίτευξη αυτών των στόχων, είναι προφανώς απαραίτητη η δυνατότητα μέτρησης των εκπομπών CO_2 από την εταιρεία. Η μέτρηση αυτή πρέπει να είναι όσο το δυνατόν κοντινότερη στην πραγματικότητα, αλλά και απολύτως πλήρης, περιλαμβάνοντας όλες τις εκπομπές, άμεσες ή έμμεσες.

Επιπροσθέτως, η Mærsk δηλώνει και ελέγχεται από ανεξάρτητη αρχή για τις ετήσιες εκπομπές CO_2. Για τη διευκόλυνση αυτού του ελέγχου, και τη διευθέτηση τυχόν ζητημάτων, απαιτείται όσο το δυνατόν μεγαλύτερη διαφάνεια και ελαχιστοποίηση ανθρώπνινων σφαλμάτων.

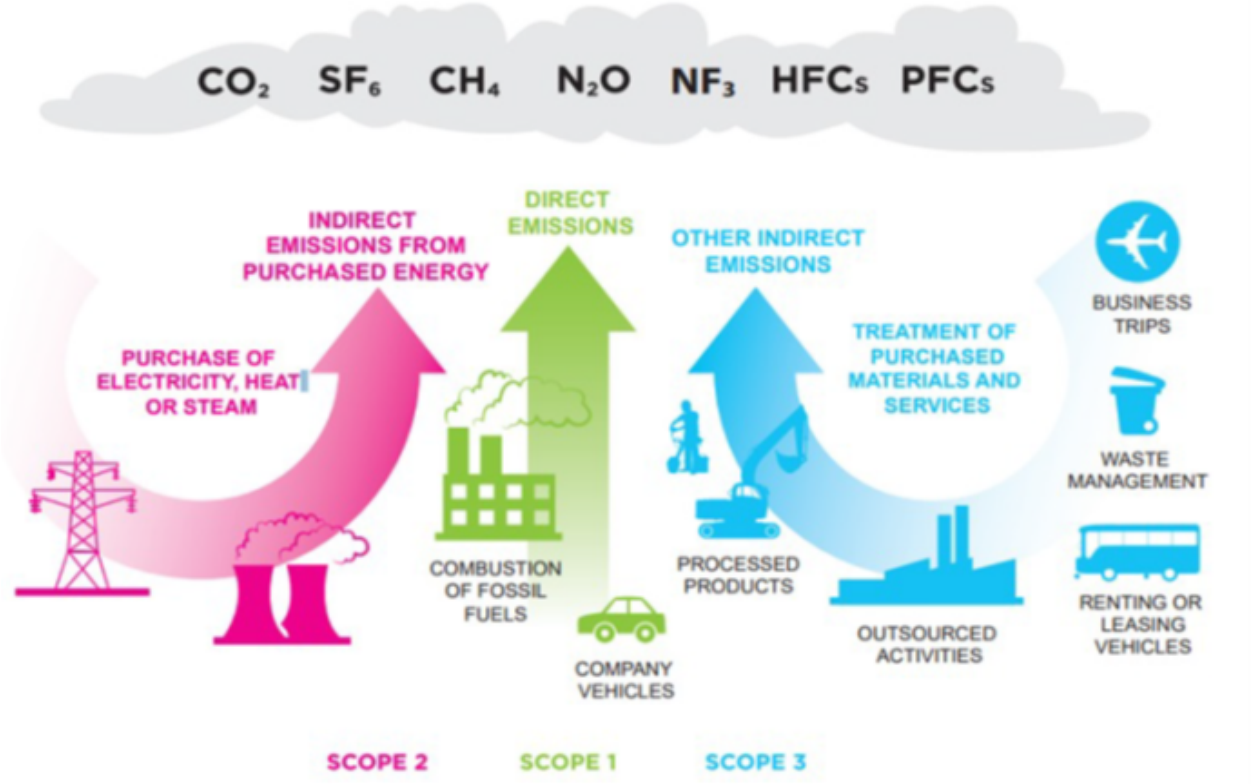
1.1 Μέτρηση εκπομπών CO_2

Οι εκπομπές CO_2 χωρίζονται συνήθως σε 3 κατηγορίες:

- Κατηγορία 1: Απ' ευθείας εκπομπές από πηγές που ανήκουν σε έναν οργανισμό (πχ καύσιμα για κίνηση πλοίων ή φορτηγών)
- Κατηγορία 2: Έμμεσες εκπομπές από την παραγωγή και προμήθεια / μεταφορά της ενέργειας που καταναλώνει ένας οργανισμός (πχ εκπομπές από εργοστάσιο

ενέργειας που ηλεκτροδοτεί τα γραφεία της εταιρείας ή μια πύλη/terminal φορτοεκφόρτωσης λιμανιού)

- Κατηγορία 3: Εκπομπές από μέσα παραγωγής που δεν ανήκουν απ' ευθείας στον οργανισμό, αλλά είναι εμμέσως υπεύθυνος για τη χρήση τους. Επιπλέον, περιλαμβάνει οποιεσδήποτε εκπομπές δεν υποπίπτουν στις κατηγορίες 1 και 2. (πχ ενοικιασμένα ρυμουλκά πλοία ή εκπομπές που συσχετίζονται με προϊόντα που αγοράζει η εταιρεία)



Σχήμα 1.1: Κατηγορίες Εκπομπών

Η μεθοδολογία με την οποία υπολογίζονται οι εκπομπές CO_2 είναι εκτός θέματος του κειμένου. Αξίζει να σημειωθεί ωστόσο, ότι υπολογίζονται με βάση διεθνώς αποδεκτή μεθοδολογία, για την οποία είναι υπεύθυνο διαφορετικό τμήμα της εταιρείας, ενώ υπόκειται σε έλεγχο από κρατικές αρχές (περισσότερα σχετικά, μετέπειτα). Ένα πολύ απλοποιημένο νοητικό μοντέλο για τις εκπομπές CO_2 αποτελεί τη μέτρηση της καταναλισκόμενης ενέργειας για την ολοκλήρωση ενός ταξιδιού, και τη μετατροπή της ενέργειας αυτής σε CO_2 (με έναν απλό πολλαπλασιασμό), με βάση έναν παράγοντα συγκεκριμένο για το είδος του καυσίμου. Για τη μέτρηση της απαιτούμενης ενέργειας μιας διαδρομής, πολύ απλά μπορεί να μετρηθεί η διαφορά στη μάζα του καυσίμου πριν και μετά τη διαδρομή, και να υπολογιστεί δεδομένου ότι είναι γνωστή η ενεργειακή απόδωση του καυσίμου ανά μονάδα μάζας.

Σαν ένα πιο απτό (και απλοποιημένο) παράδειγμα υπολογισμού: Για μια διαδρομή καταναλώνεται 1 kg πετρελαίου, με παράγοντα 3.5 gram CO_2 / gram καυσίμου, άρα

παράγονται 3.5 kg CO_2. Αν αντίθετα είχε χρησιμοποιηθεί μεθανόλη με παράγοντα 0.5 gram CO_2 / gram καυσίμου, θα είχαμε μόνο 0.5 kg CO_2, δηλαδή 86% λιγότερες εκπομπές CO_2.

1.2 Η προηγούμενη διαδικασία και τα μειονεκτήματά της

Πριν τη δημιουργία του λογισμικού του Πάγκου Εργασίας, οποιαδήποτε πληροφοριακή ανάγκη σχετική με εκπομπές CO_2 ήταν μια χρονοβόρα, χειροκίνητη διαδικασία. Δεδομένα από διαφορετικές πηγές, όπως βάσεις δεδομένων και εξειδικευμένα πληροφοριακά συστήματα, συλλέγονταν και κανονικοποιούνταν από κάποιον data analyst. Η κανονικοποίηση (πχ η ίδια τοποθεσία μπορούσε να είναι γραμμένη με διαφορετικό τρόπο σε δύο βάσεις δεδομένων) γινόταν επίσης με το χέρι, κάτι που μπορούσε να οδηγήσει σε λάθη. Τα δεδομένα αυτά εξάγονταν σε αρχεία Excel και δίνονταν για επεξεργασία στο τμήμα υπεύθυνο για τη μεθοδολογία μετρήσεων εκπομπών CO_2. Τελικά έφταναν (μετά από βδομάδες ή μήνες) στον αρχικό αιτούντα.

Αυτή η διαδικασία, εκτός απ' τα πολλά σημεία στα οποία επέτρεπε ανθρώπινο σφάλμα, ήταν εντελώς ακατάλληλη για κοινοποίηση στις ελεγκτικές αρχές. Τα δεδομένα μπορεί να είχαν αλλάξει αρκετά απ' τη στιγμή που συλλέχθηκαν μέχρι την ολοκλήρωση των υπολογισμών. Επίσης ήταν αδύνατο να εγγυηθεί κανείς πως η μεθοδολογία υπολογισμών ακολουθήθηκε ακριβώς, καθώς ήταν ένας χειρικίνητος υπολογισμός στο Excel. Τέλος, αν η εκάστοτε αρχή ήθελε να επαναλάβει κάποιον υπολογισμό για το περασμένο έτος, μπορεί τα δεδομένα να είχαν χαθεί (να είχαν αντικατασταθεί με πιο πρόσφατα).

1.3 Το λογισμικό μιας ολοκληρωμένης λύσης

Η λύση που σχεδιάστηκε και υλοποιήθηκε, αντιμετώπισε όλα τα προαναφερθέντα προβλήματα, και επέτρεψε δυνατότητες που λειτουργούν ως ανταγωνιστικό πλεονέκτημα της εταιρείας, έναντι ανταγωνιστών.

Το λογισμικό παρουσιάζεται στο χρήστη (υπάλληλοι της εταιρείας μόνο), μέσω μιας ιστοσελίδας. Εκεί ανά πάσα στιγμή υπάρχουν τα νεότερα δεδομένα που έχουμε στη διάθεσή μας, και ασύγχρονα ενημερώνεται η σελίδα ακόμα και αν είναι ήδη ανοιχτή χωρίς να χρειάζεται ανανέωση.

Όλες οι δυνατές πηγές δεδομένων (βάσεις, ροές, εξωτερικά APIs) συγκεντρώνονται και κανονικοποιούνται σε ένα σύστημα, το οποίο εκτελεί τους υπολογισμούς εκπομπών σύμφωνα με την καθορισμένη μεθοδολογία. Η συγκέντρωση της πληροφορίας, επιτρέπει επίσης την δημιουργία καταγραφών αλλαγής των δεδομένων μέσα στο χρόνο, με αποτέλεσμα τη δυνατότητα να μπορούμε ανά πάσα στιγμή να δώσουμε τόσο τα πηγαία δεδομένα, όσο και την τότε μεθοδολογία υπολογισμού στις ελεγκτικές αρχές, ώστε να μπορούν να διασταυρώσουν τα αποτελέσματά μας.

Η συνεχής ανανέωση των δεδομένων, διευκολύνει επίσης τις πωλήσεις "οικολογικών προϊόντων" (ECO Products), που αφορούν τρόπους αποστολής εμπορευμάτων με

εναλλακτικά καύσιμα, με μειωμένες εκπομπές ρύπων. Συγκεκριμένα, ανά πάσα στιγμή, ένας πωλητής μπορεί να γνωρίζει τι διαθεσιμότητα υπάρχει για κάθε εναλλακτικό καύσιμο, ποιά είναι η τρέχουσα τιμή του, κτλ ώστε να το χρεώσει αντίστοιχα.

Επιπροσθέτως, δίνεται η δυνατότητα στην εταιρεία να πουλάει “εικονικά” χαμηλότερες εκπομπές CO_2, με αποδείξιμο τρόπο ότι αυτές συνέβησαν. Για παράδειγμα, μπορεί ένας πελάτης που θέλει να μειώσει τις εκπομπές του, να πληρώσει επιπλέον ώστε να χρησιμοποιηθεί κάποιο εναλλακτικό καύσιμο. Ωστόσο, μπορεί να μην αλλάξει τίποτα στη μεταφορά των εμπορευμάτων του. Παράλληλα όμως, μπορεί ένα άλλο φορτίο σε διαδρομή που θα οδηγούσε στην ίση κατανάλωση ενέργειας, να σταλεί με εναλλακτικά καύσιμα (χωρίς να έχει πληρώσει γι’ αυτό ο ιδιοκτήτης του), και έτσι να ισοζυγιστεί από εκεί η συνολική εκπομπή CO_2. Αυτό τώρα μπορεί να είναι αποδείξιμα σωστό, καθώς το σύστημα συγκεντρώνει όλη τη χρήση καυσίμων της εταιρείας, καθώς και τα αποδεικτικά για την προμήθεια και κατανάλωση αυτών των καυσίμων (συνεπώς μπορεί να αποδειχθεί ότι συνολικά καταναλώθηκαν τα ζητούμενα kg εναλλακτικού καυσίμου, ακόμα κι αν ήταν σε διαφορετικό φορτίο).

Τέλος, ένα μεγάλο ανταγωνιστικό πλεονέκτημα αφορά την έκδοση πιστοποιητικών. Μιας και η διαδικασία είναι πλήρως ελέγξιμη, είναι δυνατό να εκδοθούν πιστοποιητικά για τις εκπομπές κάποιου πελάτη για τις συναλλαγές του με την εταιρεία. Αυτά τα πιστοποιητικά μπορούν να αποσταλούν στις αντίστοιχες αρχές, οι οποίες μπορούν να τα επιβεβαιώσουν μετέπειτα. Επίσης η έκδοση ή ανανέωση/επανέκδοση ενός πιστοποιητικού μπορεί να γίνει αυτόματα (αν πχ μετά από μερικούς μήνες έχει αλλάξει σημαντικά κάποια τιμή).

1.4 Περιορισμοί

Αυτή τη στιγμή περιλαμβάνονται μόνο μετρήσεις κατηγορίας 1 και 3. Οι μετρήσεις κατηγορίας 2 θα υλοποιηθούν μέσα στο 2025.

Κεφάλαιο 2

Τεχνολογία

Η φύση του προβλήματος ταιριάζει με φυσικό τρόπο στο νοητικό μοντέλο ενός συστήματος προμήθειας γεγονότων (event sourcing). Συγκεκριμένα, διάφορες πηγές, σύγχρονες (βάσεις δεδομένων ή APIs) ή ασύγχρονες (ουρές - queues ή ροές - data streams), περνούν από ένα pipeline (γραμμή αγωγών) δεδομένων, μετασχηματίζονται, και συλλέγονται. Όλη αυτή η επεξεργασία δεν έχει χρονικές εξαρτήσεις, συνεπώς μπορεί να παραλληλοποιηθεί πλήρως, άρα ένα σύστημα με εύκολη παραλληλία είναι επιθυμητό (horizontal scaling).

Σημαντική ανάγκη είναι η ανοχή σε σφάλματα. Μιας και το pipeline των δεδομένων έχει πολλά σημεία στα οποία μπορεί να στηριχθεί σε κλήσεις σε εξωτερικά συστήματα, η πιθανότητα για κάποιο αναπάντεχο σφάλμα εκτός του ελέγχου μας δεν πρέπει να οδηγεί σε κατάρρευση του δικού μας συστήματος ή καταστροφή δεδομένων.

2.1 Επιλογή τεχνολογιών

Η κύρια τεχνολογία που επιλέχθηκε γιατί ακριβώς ταίριαζε στις παραπάνω ανάγκες είναι η virtual machine (εικονική μηχανή) BEAM και κατ' επέκταση οι γλώσσες προγραμματισμού που τρέχουν πάνω σ' αυτή.

Η virtual machine αυτή, δημιουργήθηκε από μηχανικούς της Ericsson για την εκτέλεση της γλώσσας Erlang. Προσφέρει φτηνή και εύκολη παραλληλία τόσο σε επίπεδο διεργασίας όσο και κατανεμημένου συστήματος και ανοχή στα σφάλματα. Συγκεκριμένα, επιλέχθηκε η γλώσσα προγραμματισμού Elixir, μιας και το συντακτικό της ήταν πιο εύκολο από της Erlang, ενώ έχει αρκετή ωριμότητα (πχ πλήθος πακέτων ανοιχτού κώδικα), καθώς και πρόσβαση στο Phoenix Framework.

Για την κατασκευή της ιστοσελίδας και της διεπαφής με το χρήστη, επιλέχθηκε το framework Phoenix (παρόμοιο με το framework Ruby on Rails σε φιλοσοφία). Επιλέχθηκε καθώς επιτρέπει γρήγορη δημιουργία νέων σελίδων, αλλά και την παραγωγή πλήρως ολοκληρωμένης ροής δεδομένων από τη βάση μέχρι την παρουσίαση σε HTML, με εργαλεία γραμμής εντολών.

Επιπλέον, το Phoenix Framework παρέχει την τεχνολογία LiveView η οποία επιτρέπει στον web server να στέλνει ενημερώσεις ασύγχρονα στον περιηγητή του χρήστη, ανά πάσα στιγμή αλλάζει κάτι στα δεδομένα. Μ' αυτή την τεχνολογία μπορείς να παρέχεις μια εμπειρία όπως signle page applications, χωρίς να έχεις χωριστή βάση κώδικα για το backend και το frontend (δηλαδή τον κώδικα που τρέχει στο server και στον περιηγητή του χρήστη αντίστοιχα).

Το σύστημα πρέπει να μπορεί να ξαναστηθεί από το μηδέν, με αυτοματοποιημένο τρόπο, σε περίπτωση πλήρους καταστροφής του κέντρου στο οποίο τρέχει. Γι' αυτό το σκοπό, όλη η αρχιτεκτονική των επιμέρους τμημάτων του συστήματος περιγράφεται σε αρχεία Terraform. Τα αρχεία αυτά μπορούν να δημιουργήσουν από το μηδέν, ένα δίκτυο συστημάτων (βάσεις δεδομένων, web servers, servers observabilty, κτλ) με ντετερμινιστικό αποτέλεσμα.

Όλα τα συστήματα τρέχουν στο νέφος της Microsoft (Azure), αντί να χρειάζεται να τα διαχειριζόμαστε εμείς. Αυτό έχει το πλεονέκτημα εύκολων αναβαθμίσεων, δυνητικά αυξημένης ασφάλειας (δεδομένου ότι δεν έχει γίνει κακή παραμετροποίηση), αλλά και τη δυνατότητα να δημιουργούμε κατά βούληση νέα μηχανήματα, ή αντίγραφα περιβάλλοντα για δοκιμές. Το μειονέκτημα είναι φυσικά το μεγαλύτερο κόστος σε σχέση με υλικό που διαχειρίζεται η ίδια η εταιρεία.

Τέλος για τη δυνατότητα οριζόντιας παραλληλίας (horizontal scaling) σε επίπεδο μηχανημάτων, αλλά και της αυξημένης ανοχής σε σφάλματα, χρησιμοποιούνται εικονικά μηχανήματα διαχειριζόμενα με το λογισμικό Kubernetes, που απαρτίζουν ένα κατανεμημένο σύστημα. Συνεπώς αν ένα μηχάνημα δεν είναι διαθέσιμο λόγω σφάλματος ή αναβάθμισης, τα υπόλοιπα μπορούν να συνεχίσουν να εξυπηρετούν τους χρήστες. Επιπλέον το Kubernetes επιτρέπει τη ρύθμιση του αριθμού των αντιγράφων μηχανημάτων που χρειάζονται, τον περιορισμό των συνδέσεων προς αυτά στο ελάχιστο (για λόγους ασφαλείας), καθώς και την εύκολη ρύθμιση των τεχνικών προδιαγραφών των μηαχνημάτων (πυρήνες ΚΜΕ, μέγεθος μνήμης τυχαίας προσπέλασης).

2.1.1 Πλεονεκτήματα BEAM - Elixir

Στην virtual machine της Elixir, η παραλληλία εντός συστήματος είναι πολύ ευκολότερη απ' τη δημιουργία και διαχείριση νημάτων (threads) σε επίπεδο λειτουργικού. Συγκεκριμένα, κάθε επεξεργαστική ροή, αποτελεί ένα BEAM process (διεργασία). Στην πραγματικότητα η BEAM τρέχει σε τόσα νήματα όσα και η πυρήνες του επεξεργαστή του συστήματος, όμως μπορεί να έχει εκατοντάδες χιλιάδες BEAM processes να τρέχουν ταυτόχρονα (concurrently - όχι παράλληλα). Αυτό το πετυχαίνει με ένα δικό της σύστημα χρονοπρογραμματισμού (scheduler).

Τα BEAM processes έχουν ελάχιστο overhead δημιουργίας, πολύ χαμηλότερο ενός νήματος, και κατ' επέκταση η δημιουργία ενός για κάθε νέο δεδομένο που εισάγεται στο σύστημα είναι πολύ φτηνή. Αυτό επίσης επιτρέπει την απομόνωση της επεξεργασίας του κάθε νέου δεδομένου. Οτιδήποτε συμβεί κατά τη διάρκεια της επεξεργασίας του, δεν επηρεάζει τις άλλες εικονικές διεργασίες. Συνεπώς σε περίπτωση σφάλματος, όλα

συνεχίζουν κανονικά, και το προβληματικό δεδομένο θα μπορεί να αρχίσει απ' την αρχή την επεξεργασία του, και πιθανώς να την ολοκληρώσει επιτυχώς (αν το σφάλμα οφειλόταν σε παροδική αιτία).

Η Elixir είναι μια συναρτησιακή γλώσσα, που σημαίνει ότι φυσικά, δημιουργεί κανείς pipelines επεξεργασίας δεδομένων, ακριβώς όπως και στο νοητικό μοντέλο που αντικατοπτρίζει το σύστημά μας. Αυτό έχει το πλεονέκτημα του ότι δεν υπάρχει πρακτικά state (κατάσταση) που πρέπει να συγχρονιστεί ή διασωθεί σε περίπτωση σφάλματος. Το μεγαλύτερο μέρος του pipeline επεξεργασίας, αποτελείται από pure functions (συναρτήσεις χωρίς παρενέργειες - side effects), δηλαδή χωρίς κανένα κρυμμένο state - όσες φορές κι αν κληθούν με τα ίδια ορίσματα, παράηουν τα ίδια αποτελέσματα. Μόνο στην αρχή (εισροή δεδομένων) και στο τέλος (αποθήκευση αποτελεσμάτων) βγαίνουμε απ' τον κόσμο των pure functions, περιορίζοντας πολύ σημαντικά τα σημεία που το σύστημα εξαρτάται από εξωγενής παράγοντες (που μπορεί να παράγουν σφάλματα εκτός του ελέγχου μας).

Επειδή η BEAM σχεδιάστηκε να μπορεί να ανακάμπτει αυτόματα από σφάλματα, όλες οι διεργασίες που τρέχουν πάνω της, επανεκκινούνται αυτόματα σε περίπτωση σφάλματος. Επίσης, υπάρχει η δυνατότητα, οι διεργασίες να εκτελούνται σε διαφορετικά μηχανήματα, και να δημιουργούν ένα κατανεμημένο σύστημα. Συνήθως τα κατανεμημένα συστήματα οδηγούν σε έκρηξη πολυπλοκότητας, με συνέπεια να αποφεύγονται εκτός αν είναι απολύτως απαραίτητα. Τα BEAM processes ωστόσο είναι εξαιρετικά απλό να συμμετάσχουν στο ίδιο κατανεμημένο σύστημα, και αν χρειαστεί να επικοινωνήσουν μεταξύ τους με μυνήματα. Συγκεκριμένα το BEAM διαχειρίζεται με τον ίδιο τρόπο διεργασίες που τρέχουν στο ίδιο μηχάνημα, με άλλες που τρέχουν σε διαφορετικά με εντελώς όμοιο τρόπο, από μεριάς του προγραμματιστή. Αυτό κάνει σαφώς απλούστερη την οριζόντια παραλληλοποίηση σε περισσότερους επεξεργαστές.

2.2 Αρχιτεκτονική

Το σύστημα αποτελείται από έναν server (εξυπηρετητή) βάσης δεδομένων, με πολλές επιμέρους βάσεις δεδομένων επάνω του. Η κάθε βάση αφορά τις ανάγκες κάθε προιόντος που προσφέρεται στην πλατφόρμα του Πάγκου Εργασίας Εκπομπών.

Ο web server τρέχει σε τρία αντίγραφα (διαχειριζόμενα μέσω Kubernetes) τα οποία συνδέονται αυτόματα (μέσω ενός load balancer και reverse proxy της Akamai) με τους περιηγητές των χρηστών.

Η ιστοσελίδα απαιτεί αυθεντικοποίηση του χρήστη μέσω κλήσης SAML SSO (single sign on) στο Azure IAM (Identity Access Management). Το τελευταίο είναι ρυθμισμένο ώστε να επιτρέπει σε συγκεκριμένους χρήστες και Active Directory groups την πρόσβαση στην εφαρμογή.

Η διεργασία επεξεργασίας δεδομένων τρέχει συνεχώς (ως χωριστή διεργασία απ' τον web server) σε ένα απ' τα (εικονικά) μηχανήματα που τρέχουν οι web servers. Εξετάζει σύγχρονα εσωτερικές βάσεις δεδομένων (polling) για ενημερώσεις και εισάγει τα νέα δεδομένα εάν υπάρχουν. Παράλληλα, δέχεται ασύγχρονα δεδομένα από ουρές

Kafka (λογισμικό του Apache foundation που λειτουργεί ως message broker - μεσίτης μυνημάτων) και άλλα ιδιωτικά push APIs.

Μαζί με αυτά, υπάρχει μια διεργασία observability, δηλαδή συλλογής δεδομένων για την κατάσταση του συστήματος, η οποία είναι σε θέση να παράγει και ενημερώσεις σε περίπτωση που ανιχνευθεί κάποια βλάβη (πχ σταματήσει η εισροή δεδομένων ή ο δίσκος της βάσης δεδομένων κοντεύει να γεμίσει).

Κεφάλαιο 3

Εργασιακές μέθοδοι

3.1 Ανάπτυξη λογισμικού με μεθόδους Agile - Extreme Programming (XP)

Η ομάδα ανάπτυξης του Πάγκου Εργασίας Εκπομπών ακολουθεί την μέθοδο Extreme Programming (XP). Αυτή η μέθοδος ανάπτυξης περιλαμβάνει:

- pair programming (προγραμματισμός σε ζευγάρια)
- test driven development (ανάπτυξη οδηγούμενη από τεστς)
- continuous integration (συνεχής ενσωμάτωση κώδικα) / continuous delivery (συνεχής παράδοση)
- κατακόρυφη ιδιοκτησία κώδικα και πόρων
- μικρούς και συνεχής βρόγχους ανατροφοδότησης / σχολίων

3.1.1 Pair Programming

Στοχεύουμε σε pair programming το 100% του χρόνου (όσο είναι δυνατόν και θεμιτό). Ουσιαστικά δύο προγραμματιστές γράφουν μαζί κώδικα, ο ένας σε ρόλο “οδηγού” και ο άλλος σε ρόλο “πλοηγού”. Ο πλοηγός έχει την ευθύνη της αφηρημένης επίλυσης του προβλήματος (σε αλγοριθμικό επίπεδο, ανεξαρτήτως τεχνολογίας και γλώσσας προγραμματισμού), ενώ ο οδηγός παίρνει πιο “τακτικές” αποφάσεις, χαμηλότερου επιπέδου, όπως ποιά στοιχεία της γλώσσας προγραμματισμού αναπαριστούν καλύτερα την επιθυμητή δομή δεδομένων, ή πχ το πως να κληθεί σωστά μια εξωτερική συνάρτηση. Ο ρόλος του πλοηγού και οδηγού συνήθως αλλάζει συχνά μέσα στη μέρα, από μια φορά κάθε μερικές ώρες, εώς κάθε λίγα λεπτά, ανάλογα με τις ανάγκες του ζεύγους των προγραμματιστών.

Μια συχνή κριτική / ανησυχία σχετικά με το pair programming είναι η “σπατάλη” χρόνου. Έχουμε συγκρίνει (και σε προηγούμενη ομάδα που επίσης ακολουθούσε extreme programming), την απόδωση της ομάδας μέσα σε μερικούς μήνες με ή χωρίς pair programming, και έχουμε δει ότι παραδόξως συντέλεσε σε αύξηση της αποδοτικότητας

(μετρώντας πόσα user stories ολοκληρώθηκαν ανά μέρα σε παρόμοια διαστήματα). Αυτό, εμπειρικά, εκτιμώ πως οφείλεται πολλές φορές στην άμεση επικοινωνία, που οδηγεί τόσο σε διασπορά της γνώσης του προβλήματος, αλλά και αλληλοκάλυψη των προγραμματιστών με εξειδίκευση σε διαφορετικά πεδία (πχ κάποιος μπορεί να έχει περισσότερες γνώσεις SQL και κάποιος περισσότερες σχετικές με CSS).

Ένα απ' τα πλεονεκτήματα του pair programming είναι η αύξηση της ποιότητας του παραγόμενου κώδικα, καθώς περνά από μια διαρκή διαδικασία επισκόπησης από τον άλλο προγραμματιστή στο ζευγάρι. Συνεπώς είναι πολύ πιθανότερο να προληφθούν λάθη, αλλά και να προταθεί μια καλύτερη, πιο κατανοητή μορφή κώδικα με το ίδιο αποτέλεσμα.

Όπως αναφέρθηκε πρωτύτερα, άλλο πλεονέκτημα είναι ο διαμοιρασμός της γνώσης γύρω απ' το πραγματικό πρόβλημα που επιλύεται (πχ ορολογία ή λεπτομέρειες σχετικά με εξαιρέσεις και περιορισμούς), κάτι ελαχιστοποιεί την ανάγκη για συναντήσεις "συντονισμού" γνώσεων.

Τέλος, απλά λόγω του γεγονότος ότι δύο άτομα έχουν δημιουργήσει κάθε μέρος της βάσης κώδικα (και δεδομένου ότι τα ζεύγη αλλάζουν συχνά αν όχι καθημερινά), αποφεύγεται η δημιουργία "σιλό" γνώσεων. Επεξηγώντας, ανά πάσα στιγμή τουλάχιστον δύο άτομα στην ομάδα έχουν εμπειρία για κάθε κομμάτι κώδικα (συνήθως περισσότερα αν όχι όλα), με συνέπεια να αποφεύγεται η δημιουργία τμημάτων κώδικα άγνωστα στους προγραμματιστές που τα συντηρούν. Αυτό έχει το εξαιρετικό πλεονέκτημα του ότι τα μέλη της ομάδας είναι εύκολο να αντικατασταθούν (πχ να μοιράζονται προγραμματιστές με άλλες ομάδες) και φυσικά όταν κάποιος λείπει για οποιοδήποτε λόγο, δεν δημιουργείται σοβαρό εμπόδιο στην πρόοδο ανάπτυξης του κώδικα.

3.1.2 TDD - Test Driven Development

Η τεχνική του TDD αφορά την προσθήκη του ελάχιστου δυνατού κώδικα που πληρεί κάποια προδιαγραφή. Πρώτα εκφράζεται η προδιαγραφή, συνήθως με τη μορφή ενός αυτοματοποιημένου τεστ, και έπειτα γράφεται ο ελάχιστος απαραίτητος κώδικας ώστε να ικανοποιηθεί η απαίτηση.

Τα πλεονεκτήματα αυτής της τεχνικής είναι πολλά. Αρχικά, ως άμεσο επακόλουθο, η βάση του κώδικα έχει μια πλήρη σουίτα από αυτοματοποιημένα τεστ, για κάθε χαρακτηριστικό που θεωρείται προδιαγραφή του τελικού συστήματος. Κατά συνέπεια, οποιαδήποτε αλλαγή στον κώδικα, μπορεί εύκολα να ελεγχθεί για τυχόν παλινδρόμηση (regression) - δηλαδή ανεπιθύμητες αλλαγές στη συμπεριφορά του συστήματος. Επίσης, ακόμα και κάποιος με μικρή γνώση του συστήματος, μπορεί να έχει μεγάλη αυτοπεποίηθηση πως οι αλλαγές του δεν οδήγησαν σε σφάλματα σε άλλα σημεία.

Ένα άλλο πλεονέκτημα, είναι πως είναι εξαιρετικά ξεκάθαρο το ποιές είναι οι προδιαγραφές που πληρεί το σύστημα. Αυτό μπορεί να λειτουργήσει πολύ καλύτερα από σχόλια (comments) για την τεκμηρίωση του κώδικα, γιατί τα τεστ είναι συνυφασμένα με τον κώδικα, ενώ αντίθετα τα σχόλια μπορεί να αποκλίνουν (πχ να αλλάξει ο κώδικας χωρίς να ενημερωθούν τα σχόλια). Παράλληλα, είναι έυκολο σε μια αναθεώρηση παλαιότερου κώδικα, να κρίνουμε αν μια συμπεριφορά είναι ακόμη επιθυμητή ή θα έπρεπε να

καταργηθεί (πχ μια προδιαγραφή μπορεί να μην χρειάζεται πια, άρα το αντίστοιχο τεστ και ο κώδικας που ελέγχει είναι πολύ εύκολο να ανακαλυφθούν και να αναθεωρηθούν).

Τέλος, είναι πολύ σύνηθες να έχουμε ως αποτέλεσμα την υψηλότερη ποιότητα κώδικα ακολουθώντας TDD. Αυτό συμβαίνει καθώς αν τηρούμε τον περιορισμό του να γράφεται ο ελάχιστος κώδικας που πληρεί μια προδιαγραφή, οδηγούμαστε συνήθως σε λιγότερες εξωτερικές εξαρτήσεις (dependencies), κώδικα που είναι υπεύθυνος για ένα και μόνο στόχο, ο οποίος είναι ευκολότερο να συντηρηθεί στο μέλλον. Οι εξαρτήσεις από εξωτερικές πηγές δεδομένων γίνονται εμφανείς (καθώς πρέπει να ελέγχονται από το τεστ), κάτι που οδηγεί σε αρθρωτό κώδικα που εύκολα αντικαθίσταται ή μεταφέρεται.

3.1.3 CI / CD - Continuous Integration / Continuous Delivery

Το CI (συνεχής ενσωμάτωση) αφορά αυτοματοποιημένο κύκλο ενσωμάτωσης κώδικα και δημοσίευσης ενημερώσεων του συστήματος. Ακολουθώντας αυτήν την πρακτική, κάθε μικρή αλλαγή κυκλοφορεί άμεσα, κάτι που δίνει πληροφορίες στον προγραμματιστή, τόσο από το ίδιο το σύστημα, όσο και απ' τους χρήστες του.

Κάτι τέτοιο επιτυγχάνεται με τη χρήση συστήματος διαχείρισης εκδόσεων κώδικας (VCS - Version Control System), που στην περίπτωσή μας έχει επιλεγεί το git. Συγκεκριμένα τηρούμε μόνο ένα branch (κλάδο) και όλες οι αλλαγές γίνονται απ' ευθείας εκεί. Επιπλέον, δημοσιοποιούνται όσο πιο άμεσα γίνεται, πολλές φορές τη μέρα. Αυτό έχει ως αποτέλεσμα, ανά πάσα στιγμή όλοι οι προγραμματιστές να έχουν την πλέον πρόσφατη έκδοση του συστήματος, αποφεύγοντας περίπλοκες συνδιαλλαγές διαφορετικών κλάδων κώδικα που έχουν αποκλίνει επί μέρες.

Ένα άλλο επακόλουθο είναι πως οι αλλαγές αυτές φτάνουν και στο χρήστη, άμεσα, κάτι που δίνει τη δυνατότητα για σχόλια σχετικά με την κατεύθυνση ανάπτυξης του προϊόντος (πχ αν εξυπηρετεί τις ανάγκες του, ή είναι εύχρηστο).

Για την υλοποίηση της αυτοματοποιημένης δημοσίευσης κάθε νέας έκδοσης του συστήματος, χρησιμοποιείται το σύστημα Github Actions. Στην τρέχουσα ομάδα 20 προγραμματιστών έχουμε κατά μέσο όρο 32 δημοσιεύσεις ενημερώσεων ανά ημέρα.

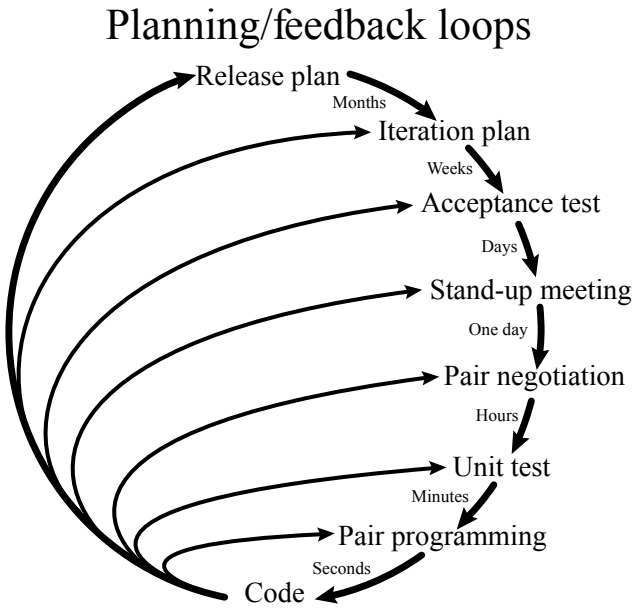
3.1.4 Vertical Ownership (κατακόρυφη ιδιοκτησία)

Στόχος της ομάδας είναι η πλήρης ιδιοκτησία της λύσης λογισμικού την οποία αναπτύσσει. Συγκεκριμένα είμαστε υπεύθυνοι για: * την ρύθμιση του υλικού (εικονικού στο νέφος της Microsoft) * την παραμετροποίηση συστημάτων δικτύου και ασφαλείας * την αποθήκευση δεδομένων (πχ σχεδιασμός βάσεων, κανονικοποίηση, κτλ) * την διαχείριση της βάσης του κώδικα * τον έλεγχο καλής λειτουργίας του συστήματος (κυρίως μέσω αυτοματοποιημένων τεστ και τηλεμετρίας) * τη συλλογή στατιστικών και στοιχείων τηλεμετρίας (πχ σχετικά με την υγεία του συστήματος) * το σχεδιασμό και υλοποίηση της διεπαφής του χρήστη

Η πρακτική αυτή έχει τα εξής πλεονεκτήματα: * μεγάλη ευελιξία και ταχύτητα για αλλαγές * αυτονομία ως προς εξωτερικές ομάδες * ελαχιστοποίηση πολυπλοκότητας κάθε

συστήματος ανάλογα με τις ανάγκες της ομάδας * δυνατότητα επίλυσης σχεδόν κάθε προβλήματος, χωρίς τη διαμεσολάβηση τρίτων

3.1.5 Feedback loops (βρόγχοι ανατροφοδότησης)



Σχήμα 3.1: Βρόγχοι Ανατροφοδότησης XP

Για κάθε επιτυχημένο εγχείρημα, είναι απαραίτητο να υπάρχει ροή πληροφορίας από τον εκάστοτε ενδιαφερόμενο / χρήστη, πίσω στο δημιουργό του συστήματος. Αυτό δίνει τη δυνατότητα στην ομάδα ανάπτυξης να γνωρίζει εάν βρίσκεται στο σωστό δρόμο ή πρέπει να αναπροσαρμόσει την πορεία της για να καλύψει τις ανάγκες των χρηστών.

Στα πλαίσια του XP, δίνεται ιδιαίτερη σημασία στο χτίσιμο πολλαπλών, όσο το δυνατό συντομότερων βρόγχων ανατροφοδότησης. Ο μικρότερος απ' αυτούς είναι η συζήτηση εντός του ζεύγους προγραμματιστών, ανατροφοδότηση που επαναλαμβάνεται κάθε μερικά δευτερόλεπτα. Έπειτα, πληροφορία από τα τεστς και τον μεταγλωττιστή κώδικα, κάθε μερικά λεπτά. Σε ημερήσια κλίμακα έχουμε μια καθημερινή συνάντηση (daily stand up), ενώ σε κλίμακα ημερών, πληροφορία από τεστ αποδοχής (acceptance tests) συνήθως από αντιπροσώπους χρηστών. Πέραν αυτών, σε επίπεδο εβδομάδων οργανώνουμε το επόμενο βήμα, κατανέμοντας μεγαλύτερες απαιτήσεις σε μικρότερα παραδοτέα, ενώ σε επίπεδο μηνών έχουμε πιο αφηρημένα στρατηγικά πλάνα που ορίζουν την κατεύθυνση που θέλουμε να εξυπηρετήσουμε στο μέλλον.

Αξίζει να σημειωθεί πως στην πραγματικότητα, η συλλογή σχολίων είναι λιγότερο δομημένη και πολύ πιο ασύγχρονη απ' ότι παρουσιάζεται παραπάνω. Επειδή παραδίδουμε πολλές φορές μέσα στη μέρα ενημερώσεις στους χρήστες μας, αυτοί με τη σειρά τους μπορούν να μας ενημερώσουν σε περίπτωση που συναντήσουν κάποιο πρόβλημα, είτε χρηστικότητας

είτε ακρίβειας δεδομένων. Επίσης είναι συχνό να συλλέγουμε σχόλια αναφορικά με την ευχρηστία της ιστοσελίδας, και να πορευόμαστε ανάλογα προς την βελτίωσή της.

Κεφάλαιο 4

Βιβλιογραφία

Σχήμα 1. - CC BY 4.0, https://commons.wikimedia.org/w/index.php?curid=140748311

Σχήμα 2. - By DonWells - Own work based on: XP-feedback.gif, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=27448045