ΔΙΑΤΜΗΜΑΤΙΚΟ ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

ΣΤΑ ΠΛΗΡΟΦΟΡΙΑΚΑ ΣΥΣΤΗΜΑΤΑ

Διπλωματική Εργασία

ΤΕΧΝΟΛΟΓΙΕΣ ΥΠΟΛΟΓΙΣΤΙΚΗΣ ΝΕΦΟΥΣ ΑΝΕΥ ΔΙΑΚΟΜΙΣΤΩΝ ΚΑΙ ΠΕΡΙΠΤΩΣΗ ΧΡΗΣΗΣ ΜΕ ΓΕΩΧΩΡΙΚΑ ΔΕΔΟΜΕΝΑ

του

ΑΛΜΠΑΝΗ-ΚΟΝΤΖΙΛΑ ΑΡΙΣΤΕΙΔΗ-ΧΡΗΣΤΟΥ

Υποβλήθηκε ως απαιτούμενο για την απόκτηση του Μεταπτυχιακού

Διπλώματος Ειδίκευσης στα Πληροφοριακά Συστήματα

Φεβρουάριος 2020

Περίληψη

Η ραγδαία ανάπτυξη του Cloud Computing (Υπολογιστική Νέφους) την τελευταία δεκαετία αποτελεί μια τεράστια πρόοδο στην υπολογιστική επιστήμη. Επιφέρει ριζικές αλλαγές στον τρόπο με τον οποίο πραγματοποιείται η υπολογιστική και ευρύτερα στον τρόπο ανάπτυξης ψηφιακών υπηρεσιών αλλά και της χρήσης αυτών. Μια από τις Cloud τεχνολογίες που φιλοδοξεί να αλλάξει άρδην τον τρόπο σύλληψης και υλοποίησης εφαρμογών ιστού είναι αυτή του Serverless Computing (Υπολογιστικής άνευ Διακομιστών).

Στόχος της παρούσας εργασίας είναι η εξέταση και αξιολόγηση των τεχνολογιών του Cloud Computing και ειδικότερα του Serverless Computing στην ανάπτυξη διαδικτυακών εφαρμογών καθώς και την διερεύνηση της συμβατότητας των πρώτων με τις τεχνολογίες των Γεωγραφικών Πληροφοριακών Συστημάτων (GIS).

Το παρόν πόνημα αποτελείται από δύο μέρη. Στο πρώτο μέρος μελετήθηκε η τεχνολογία του Cloud Computing, η εξελικτική της πορεία και η αγορά της σήμερα. Έπειτα αναλύθηκαν τα πλεονεκτήματα και μειονεκτήματα ανάπτυξης και διάταξης εφαρμογών χρησιμοποιώντας το Cloud και ειδικότερα τις τεχνικές του Serverless. Στο δεύτερο μέρος προτείνεται μια ενδεικτική υλοποίηση (PoC) μιας GIS εφαρμογής χρησιμοποιώντας τις προαναφερθείσες τεχνολογίες σε απάντηση των ερωτημάτων που διατυπώθηκαν.

Το πρωτότυπο που αναπτύχθηκε ανταποκρίθηκε στις απαιτήσεις που τέθηκαν γεγονός που υποδηλώνει το εφικτό της χρήσης Serverless πλατφορμών για την ανάπτυξη εφαρμογών που κάνουν χρήση τεχνολογιών GIS.

Λέξεις-Κλειδιά: Υπολογιστική Νέφους, Υπολογιστική άνευ Διακομιστών, Εφαρμογές Ιστού, ΓΠΣ, Python, GeoDjango

Κατηγορία Διπλωματικής: Σχεδίαση και Ανάπτυξη Λογισμικού

Abstract

The rapid developments in Cloud Computing during the last decade consist a major breakthrough for Computer Science. They carry with them radical changes to the way computing is conducted as well as to the techniques followed for the development of digital services and the ways the latter are consumed. One of the Cloud technologies of seeking to alter the way we conceive and develop web applications is Serverless Computing.

The primary goal of the current project is the investigation and evaluation of the utilization of Cloud Computing and Serverless in the development of web applications as well as the compatibility of the former with the technologies of Geographical Information Systems (GIS).

The current work is consisted of two parts. In the first part a study has been conducted concerning the technology of Cloud Computing, its progress through time and its current market status. An analysis of the benefits and drawbacks of developing and deploying applications using Cloud technologies and in particular Serverless techniques follows. In the second part an indicative implementation (PoC) of a generic GIS app is proposed utilizing the aforementioned technologies in an attempt to provide answers to the research questions posed.

The prototype developed fulfilled the requirements set proving the feasibility of utilizing Serverless platforms when developing Cloud-GIS applications.

Keywords: Cloud Computing, Serverless, Web Applications, GIS, Python, GeoDjango

ACM: D2 Software Engineering

Περιεχόμενα

Πίνακας εικονογραφήσεων

Κατάλογος Πινάκων

Κατάλογος Εικόνων

Κατάλογος Διαγραμμάτων

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον Στέργιο για τις πολύτιμες συμβουλές σχετικά με την οργάνωση της εργασίας και την Ευτέρπη για την απέραντη υπομονή της στις ατελείωτες ερωτήσεις και προβληματισμούς μου για τα ΓΠΣ.

Ένα ιδιαίτερο ευχαριστώ στην Φωτεινή για την φροντίδα, την συντροφικότητα και την υποστήριξη της. Χωρίς την συμβολή της αμφιβάλλω αν θα κατάφερνα να ολοκληρώσω την παρούσα.

Τέλος θα ήθελα να ευχαριστήσω τον καθηγητή κ. Γιώργο Ευαγγελίδη για την στήριξη και καθοδήγησή του.

1. Εισαγωγή

Η δεκαετία που διανύουμε χαρακτηρίζεται, μεταξύ άλλων και από την καθιέρωση του Cloud Computing ως την νέα νόρμα στο πεδίο των εφαρμογών της επιστήμης της πληροφορικής (Information Technology). Πλέον, η μεγάλη πλειοψηφία των οργανισμών, ιδιωτικών και δημοσίων, χρησιμοποιούν τις υπηρεσίες του Cloud αναζητώντας πιο καινοτόμες και αποδοτικές λύσεις στα πλαίσια του εν εξελίξει ψηφιακού μετασχηματισμού της οικονομίας και της κοινωνίας.

Η ευρεία υιοθέτηση του Cloud Computing οδήγησε στην αλλαγή μακροχρόνια καθιερωμένων μεθόδων στην ανάπτυξη εφαρμογών καθώς επίσης και στην ανάδυση νέων. Χαρακτηριστική περίπτωση είναι αυτή του Serverless Computing. Πρόκειται για την μέθοδο ανάπτυξης και διάθεσης διαδικτυακών εφαρμογών που αφαιρεί αρκετά από τα παραδοσιακά επίπεδα (layers) μιας εφαρμογής, επιτρέποντας έτσι την εστίαση στην επιχειρησιακή λογική βελτιστοποιώντας τους πόρους που χρησιμοποιούνται. Έτσι, ολοένα και περισσότεροι οργανισμοί στρέφονται σε Serverless τεχνικές στην προσπάθεια αναδιάταξης (ή και εκ νέου δημιουργίας) των διαδικασιών και των υπηρεσιών τους. Παρότι το Serverless Computing έχει γνωρίσει σημαντική πρόοδο τα τελευταία χρόνια και προτιμάται από έναν αυξανόμενο αριθμό χρηστών, εξακολουθεί να αποτελεί μία σχετικά νέα τεχνολογία, για την οποία υπάρχουν ακόμη αρκετά ανοιχτά ερευνητικά ζητήματα να μελετηθούν και να διερευνηθούν.

Πιο συγκεκριμένα είναι υπό διερεύνηση η καταλληλότητα και η συμβατότητα αυτής της νέας τεχνολογίας με διάφορους εμπορικούς, επιστημονικούς, βιομηχανικούς και οικονομικούς τομείς, ώστε να εκτιμηθεί η υψηλότερη δυνατή αποδοτικότητά του. Επιπρόσθετα, αποτελεί ζήτημα προς απόδειξη, η εύκολη και απλή χρήση της, σε σχέση με παρόμοιες τεχνολογικές προτάσεις και εφαρμογές, που φιλοδοξούν να καλύψουν τις ίδιες ανάγκες στην αγορά.

Σκοπός της παρούσας εργασίας είναι η ανάλυση και εξέταση της βιωσιμότητας και της πρακτικότητας της τεχνολογίας Serverless σε συνδυασμό με άλλες συμπληρωματικές τεχνολογίες, ώστε να επιλύονται σύνθετα ζητήματα διαχείρισης, ανάλυσης, απεικόνισης και αποθήκευσης διαφόρων ειδών δεδομένων. Πιο συγκεκριμένα ερευνητικός στόχος αποτελεί ο συνδυασμός Serverless τεχνολογιών με τα Γεωγραφικά Πληροφοριακά Συστήματα (GIS). Επιλέγεται η μελέτη αυτής της σχέσης λόγω του

αυξανόμενου ενδιαφέροντος και της ευρείας χρήσης γεωχωρικών δεδομένων σε πολλές εφαρμογές.

Για την επίτευξη του παραπάνω στόχου το παρόν κείμενο διαρθρώνεται σε δυο μέρη. Στο πρώτο μέρος εξετάζουμε την τεχνολογία του Cloud Computing και των GIS και στο δεύτερο μέρος παρουσιάζουμε την σχεδίαση και υλοποίηση μιας διαδικτυακής εφαρμογής βασιζόμενη στις δυο προηγούμενες τεχνολογίες.

Ως οδηγός για την ανάπτυξη της διαδικτυακής εφαρμογής, χρησιμοποιήθηκε το πρόβλημα της καταγραφής και διαχείρισης καλλιεργειών επειδή σαν θέμα συναντά πολλές από τις προκλήσεις - απαιτήσεις μιας χωρικής εφαρμογής. Ωστόσο η εφαρμογή δεν στοχεύει αποκλειστικά στο συγκεκριμένο πρόβλημα, ούτε το εξαντλεί, απλά χρησιμοποιήθηκε σαν ερέθισμα για ιδέες υλοποίησης. Επιχειρήθηκε να δημιουργηθεί ένα ανοιχτό πλαίσιο εφαρμογής σε ότι αφορά τη θεματολογία που θα αντιμετωπίζει, τα χαρακτηριστικά της, και τη δυνατότητα χρήσης της. Αυτό απλά σημαίνει ότι μπορεί να χρησιμοποιείται σε μία σειρά από περιπτώσεις που χρήζουν συλλογής και επεξεργασίας γεωγραφικών δεδομένων, από διαφορετικούς χρήστες, με διαφορετικά δικαιώματα-δυνατότητες και διαφορετική θεματολογία.

2. Cloud Computing & GIS: Επισκόπηση των Εννοιών

Σε αυτό το κεφάλαιο θα επιχειρήσουμε μια βιβλιογραφική επισκόπηση του όρου Cloud Computing, θα παραθέσουμε στοιχεία για τον κλάδο σήμερα καθώς και μια σύντομη ιστορική αναδρομή στην εξέλιξη του. Έπειτα θα εξετάσουμε στην τεχνολογία των GIS, την εξέλιξη τους στο χρόνο και την συσχέτιση τους με το Cloud.

2.1. Cloud Computing

Ο όρος Cloud Computing εμφανίζεται αρκετά συχνά σε δημοφιλείς στήλες και άρθρα για νέες τεχνολογίες δημιουργώντας έτσι την αίσθηση ότι πρόκειται για κάτι εντελώς νέο ή ίσως ακόμα ότι πρόκειται μόνο για μια εξειδικευμένη και πολύπλοκη τεχνολογία. Εάν όμως μελετήσουμε τον όρο πιο προσεκτικά θα δούμε πως δεν πρόκειται για κάτι τόσο νέο αλλά και ίσως ότι η εν λόγω έννοια δεν αναφέρεται αμιγώς σε τεχνικές λεπτομέρειες ή δεν αφορά μόνο μια τεχνολογία.

2.1.1. Ορισμός

Στην βιβλιογραφία και στα σχετικά με το θέμα επιστημονικά κείμενα η προσέγγιση της έννοιας γίνεται παραθέτοντας κάποιον ορισμό, αρκετές φορές αυτόν που δίνει ο NIST (National Institute of Standards and Technology). Ακολουθώντας αυτό το παράδειγμα λοιπόν παραθέτουμε παρακάτω τον εν λόγω ορισμό:

Το Cloud Computing είναι ένα μοντέλο το οποίο επιτρέπει την απανταχού, εύκολη και κατ' απαίτηση πρόσβαση (μέσω δικτύου) σε ένα διαμοιραζόμενο σύνολο από παραμετροποιήσιμους υπολογιστικούς πόρους (π.χ. δίκτυα, διακομιστές, αποθήκευση, εφαρμογές και υπηρεσίες) οι οποίοι μπορούν διατεθούν με μεγάλη ταχύτητα και να απελευθερωθούν με ελάχιστη διαχειριστική προσπάθεια ή χωρίς την διάδραση με τον πάροχο των πόρων. Το μοντέλο του Cloud προωθεί την διαθεσιμότητα και αποτελείται από πέντε θεμελιώδη χαρακτηριστικά, τρία μοντέλα υπηρεσιών και τέσσερα μοντέλα διάθεσης (Hogan, Liu, Sokol, & Tong, 2011). Ο παραπάνω ορισμός συμπληρώνεται από τα χαρακτηριστικά και τα μοντέλα υπηρεσιών και διάθεσης του Cloud για τα οποία γίνεται λόγος παρακάτω.

Μια ίσως απλούστερη διατύπωση του παραπάνω ορισμού είναι ότι το Cloud δεν αποτελεί τίποτε άλλο παρά ένα τεράστιο σύνολο υπολογιστικών πόρων οι οποίοι είναι

διαθέσιμοι 24 ώρες το 24ωρο, προσβάσιμοι από οποιοδήποτε σημείο στον κόσμο, χρησιμοποιώντας διαδικτυακές τεχνολογίες σε μια βάση πληρώνεις-ότι-χρησιμοποιείς (Agarwal & Saran Srivastava, 2017).

Όπως βλέπουμε οι παραπάνω ερμηνείες του όρου έχουν ως κεντρικό στοιχείο ένα μοντέλο για την παροχή, χρήση και διάθεση υπαρχόντων υπολογιστικών πόρων. Παρόμοια ερμηνεία δίνει και ένας από τους μεγάλους παρόχους Cloud Services, η IBM[1]:

Cloud Computing, ή απλά Cloud είναι η κατ' απαίτηση παράδοση υπολογιστικών πόρων – οτιδήποτε από εφαρμογές μέχρι κέντρα δεδομένων (data centers) – μέσω του διαδικτύου σε μια βάση πληρώνεις-για-την-χρήση. Ο ορισμός συνεχίζει δίνοντας έμφαση σε τρία (από τα πέντε που περιλαμβάνονται στον ορισμό του NIST) (Hogan et al., 2011) βασικά χαρακτηριστικά του Cloud:

- **Ελαστικότητα πόρων** – η εύκολη επέκταση (scale up) ή συστολή (scale down) των παρεχόμενων πόρων
- **Μετρούμενες υπηρεσίες** – Πληρωμή με βάση την χρήση
- **Self-Service** – Η εύρεση και παροχή των πόρων γίνεται απευθείας από τον πελάτη χωρίς επιπλέον αλληλεπίδραση με τον πάροχο

Τα άλλα δύο χαρακτηριστικά που αναφέρονται στον ορισμό του NIST σχετίζονται πιο άμεσα με την τεχνολογική υποδομή που καθιστά αυτό το μοντέλο της απομακρυσμένης διάθεσης πόρων εφικτό:

- **Ευρεία πρόσβαση δικτύου** – η πρόσβαση στις υπηρεσίες / πόρους γίνεται εφικτή χρησιμοποιώντας δικτυακές υποδομές και κοινώς αποδεκτούς (standard) μηχανισμούς επικοινωνίας τα οποία επιτρέπουν την χρήση ετερογενών πλατφορμών για τον πελάτη (thick or thin clients)
- **Συγκέντρωση πόρων** (resource pooling) – οι συγκεντρωμένοι πόροι χρησιμοποιούνται για την εξυπηρέτηση πολλαπλών πελατών χρησιμοποιώντας ένα μοντέλο πολλαπλών-ενοίκων (multi-tenancy) στους διαφορετικούς

[1] https://www.ibm.com/cloud/learn/cloud-computing

φυσικούς και εικονικούς πόρους οι οποίοι αναθέτονται και μεταθέτονται σε διαφορετικούς χρήστες δυναμικά ανάλογα με την εκάστοτε ζήτηση

Η καλύτερη κατανόηση του Cloud Computing γίνεται πιο εύκολα έχοντας υπόψιν την έννοια της αφαίρεσης όπως αυτή χρησιμοποιείται στην υπολογιστική επιστήμη. Με τον όρο αφαίρεση αναφερόμαστε στην διαδικασία ή διαδικασίες απομάκρυνσης φυσικών, χωρικών ή και χρονικών λεπτομερειών ή γνωρισμάτων των υπό μελέτη συστημάτων με στόχο την εστίαση σε λεπτομέρειες μεγαλύτερης (σχετικά) σημασίας. Η τεχνολογία του Cloud Computing λοιπόν επιτρέπει την παροχή υπολογιστικών πόρων στον καταναλωτή αποκρύπτοντας τις λεπτομέρειες που σχετίζονται με αυτήν διαδικασία. Με άλλα λόγια επιτυγχάνει την αφαίρεση της πολυπλοκότητας στην παροχή των πόρων.

Τα διαφορετικά μοντέλα υπηρεσιών του Cloud Computing που ακολουθούν συνήθως τις περιγραφές και προσεγγίσεις της τεχνολογίας είναι στην ουσία διαφορετικές βαθμίδες αφαίρεσης της παροχής υπολογιστικών πόρων και ομαδοποιούνται ως εξής:

1. **Infrastructure as a Service – IaaS** – αφαίρεση λεπτομερειών σε φυσικό / υλικό επίπεδο
2. **Platform as a Service – PaaS** – η παραπάνω αφαίρεση συν την αφαίρεση θεμελιωδών στοιχείων λογισμικού
3. **Software as a Service – SaaS** – παντελής αφαίρεση του υλικού και λογισμικού – πλήρης εστίαση στην επιχειρηματική λογική

Λόγω της αυξανόμενης απήχησης και χρήσης των υπηρεσιών του Cloud εμφανίστηκαν και συνεχίζουν να εμφανίζονται νέα μοντέλα με την μορφή “Something” as a Service, εξέλιξη η οποία αντικατοπτρίζει και την οικονομική τάση της προσφοράς του αποτελέσματος οποιασδήποτε παραγωγικής διαδικασίας ως υπηρεσία με στόχο την μεγιστοποίηση της αξίας για τον καταναλωτή. Η τάση αυτή είναι ευρύτερα γνωστή με τον συμβολισμό XaaS.

Έχοντας αναφέρει τα παραπάνω ίσως είναι πλέον πιο εύκολη η αντίληψη του Cloud Computing ως μια έννοια η οποία σχετίζεται μεν με τις τεχνικές εξελίξεις σε επίπεδο υλικού και λογισμικού αλλά επίσης αναφέρεται και στα οικονομικά μοντέλα αξιοποίησης αυτών. Ένας ορισμός της έννοιας από αυτήν την σκοπιά είχε γίνει από

τον Καθηγητή Ramnath Chellappa του Πανεπιστημίου του Έμορυ (Emory University) και του Πανεπιστημίου της Νότιας Καλιφόρνια (University of Southern California), σε μια ομιλία του με τίτλο "Διαμεσολαβητές στην Υπολογιστική Νέφους" (Intermediaries in Cloud-Computing), η οποία παρουσιάστηκε στο Ντάλας (Dallas) το 1997. Ο ορισμός: "Ένα μοντέλο όπου τα όρια της υπολογιστικής θα καθορίζονται από την οικονομική λογική και όχι απλά από τα τεχνολογικά όρια[2].

2.1.2. Ιστορική αναδρομή

Όπως αναφέραμε στην αρχή του κεφαλαίου και όπως ενδεχομένως θα παρατηρήσει κάποιος εάν μελετήσει την απαρχή και την εξέλιξη της εν λόγω τεχνολογίας, το Cloud Computing δεν είναι τόσο καινούργια ιδέα και έχει τις ρίζες του στην δεκαετία του 50' (Varghese, 2019).

Κατά τις δεκαετίες 50' και 60' με τον όρο υπολογιστές αναφερόμασταν σε ογκώδη και πολύ υψηλού κόστους μηχανήματα (mainframes) τα οποία συνήθως τοποθετούνταν σε ένα δωμάτιο και εκτελούσαν υπολογισμούς μέσω εντολών που λάμβαναν από τερματικά χαμηλού επιπέδου (dumb). Το 1957 ο καθηγητής John McCarthy (γνωστός για την επινόηση του όρου "τεχνητή νοημοσύνη") πραγματοποίησε την πρώτη πρόταση δημιουργίας / μετατροπής των mainframes σε χρονομεριστικά (time-sharing) μηχανήματα (McCarthy, 1992). Στα επόμενα χρόνια η ιδέα του χρονομερισμού της υπολογιστικής δύναμης των mainframes έγινε πραγματικότητα και έτσι για πρώτη φορά στην ιστορία υπολογιστικοί πόροι ενός κόμβου μοιράζονται σε πολλούς χρήστες χαρακτηριστικό του σημερινού Cloud (resource pooling – multitenancy). Ο καθηγητής McCarthy υπήρξε επίσης από τους πρώτους επιστήμονες που υποστήριζε την έννοια του Utility Computing χαρακτηριστικό συγγενές της τεχνολογίας του Cloud (metered services).

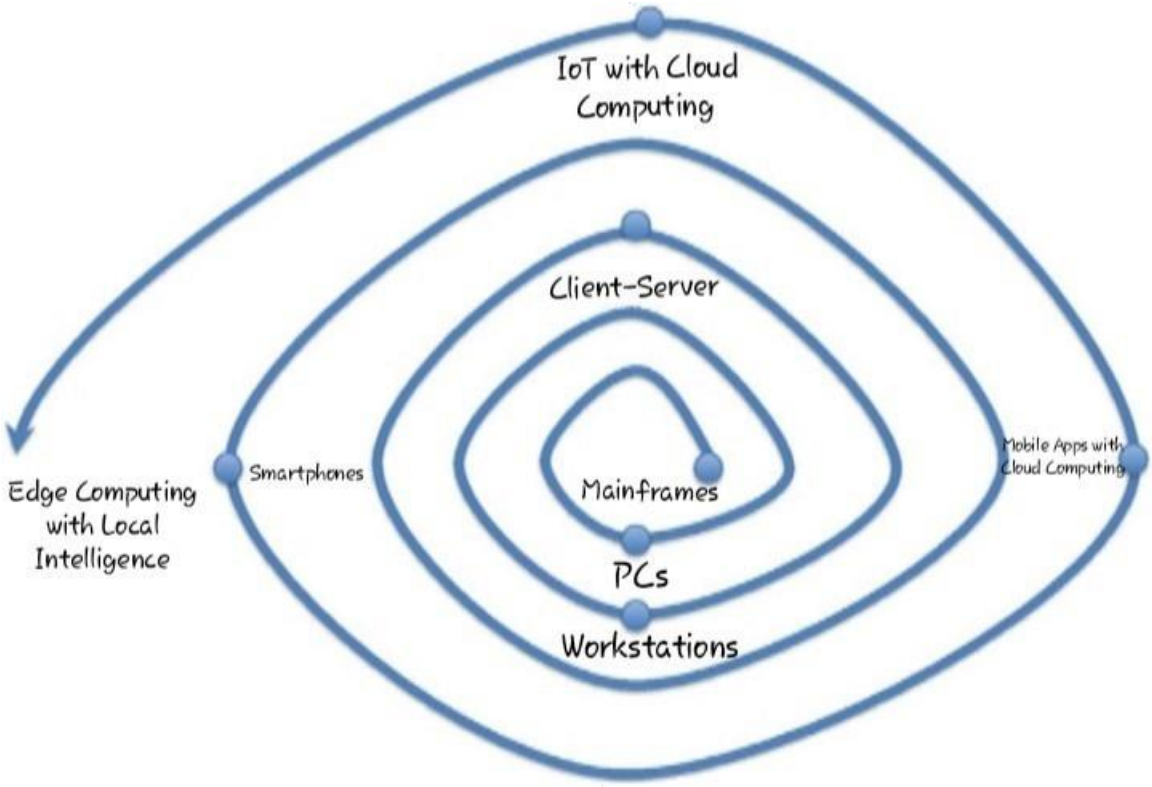
Οι επόμενες δεκαετίες (70' - 80') σημαδεύονται από δυο σημαντικές τεχνολογικές εξελίξεις: 1) την δημιουργία του προκατόχου του σημερινού διαδικτύου (ARPANET) και 2) την σχετικά ευρεία κυκλοφορία των PCs (Personal Computers). Οι τελευταίοι είχαν αρκετή ισχύ ώστε να εκτελούν βασικές εργασίες αλλά και να επικοινωνούν με άλλους απομακρυσμένους υπολογιστές, συνήθως αρκετά ισχυρότερους, κάνοντας χρήση του μοντέλου πελάτης-διακομιστής (Client-Server). Το μοντέλο αυτό

[2] https://goizueta.emory.edu/faculty/profiles/ramnath-k-chellappa

καθιερώνεται την δεκαετία του 90’ με την δημιουργία του παγκόσμιου ιστού και την άνθιση των φυλλομετρητών. Χρησιμοποιώντας τους τελευταίους οι τελικοί χρήστες χρησιμοποιούν τις “έξυπνες” συσκευές τους (PC) για να συνδεθούν στο παγκόσμιο δίκτυο και να αλληλεπιδράσουν με τους πόρους του (Sehgal & Bhatt, 2018).

Η αρχή του νέου αιώνα φέρνει την ακμή των φορητών τηλεφώνων τα οποία παρ’ όλη την μικρή ισχύ τους έχουν την δυνατότητα σύνδεσης με απομακρυσμένους υπολογιστές για τη διεκπεραίωση μιας σειράς από εργασίες και για την εξυπηρέτηση διαφόρων αναγκών ανεξαρτήτως χρόνου και χώρου. Είναι στην περίοδο πρώτης δεκαετίας που θα κάνει την εμφάνιση της η τεχνολογία του Cloud όπως την γνωρίζουμε σήμερα. Ειδικότερα το 2002 η Amazon ιδρύει την θυγατρική της Amazon Web Services με σκοπό την προσφορά υπολογιστικών πόρων και υπηρεσιών στο ευρύ κοινό. Το 2006 η υπηρεσία EC2 (Elastic Compute Cloud – VM procurement) προσφέρεται για πρώτη φορά η οποία αποτέλεσε και την βασική IaaS λύση της Amazon. Την περασμένη δεκαετία έχουμε την εμφάνιση και κυριαρχία των “έξυπνων” τηλεφώνων (smart phones) καθώς επίσης και την απαρχή του διαδικτύου των πραγμάτων (Internet of Things – IoT). Από μια πιο αφηρημένη σκοπιά έχουμε την εμφάνιση νέων πιο έξυπνων συσκευών-πελατών αλλά και μια νέα σειρά χαμηλού επιπέδου συσκευών-πελατών που όμως έχουν βασικές δυνατότητες δικτύου (Sehgal & Bhatt, 2018).

Με μια πιο προσεκτική ματιά στα στιγμιότυπα τεχνολογικής εξέλιξης που παρατέθηκαν παραπάνω μπορεί να παρατηρηθεί ένα κυκλικό μοτίβο. Ειδικότερα οι Sehgal & Bhatt (2018) υποστηρίζουν ότι η αίσθηση του κυκλικού μοτίβου δεν είναι τίποτα άλλο παρά η προβολή μια σπειροειδούς πορείας εξέλιξης (η οποία εικονίζεται παρακάτω) που ακολουθεί η επιστήμη της υπολογιστικής.



Εικόνα 1. Σπειροειδής εξέλιξη της υπολογιστικής

Η παρατήρηση σχετικά με την εξέλιξη της υπολογιστικής θεωρείται αρκετά χρήσιμη καθώς επιτρέπει την βαθύτερη κατανόηση των τεχνολογικών κινήσεων και την ευκολότερη πρόβλεψη μελλοντικών προόδων.

2.1.3. Το Cloud σήμερα

Σήμερα η αγορά του Cloud Computing είναι μια από τις πιο ραγδαία αναπτυσσόμενες παρότι έχουν ήδη διαμορφωθεί αρκετά χαρακτηριστικά της. Σύμφωνα με στοιχεία της Gartner η παγκόσμια αγορά των δημοσίων υπηρεσιών Cloud (public Cloud Services) αναμένεται να αναπτυχθεί κατά 17.5% το 2019 ανερχόμενη στα 214.3 δισ δολάρια από τα 182.4 το 2018. Στον παρακάτω πίνακα παρατίθενται ιστορικά στοιχεία και μελλοντικές εκτιμήσεις της ίδιας εταιρίας ανά κατηγορία υπηρεσιών του Cloud:

Πίνακας 1. Προβλέψεις εσόδων για τις δημόσιες υπηρεσίες Cloud παγκοσμίως (δισ. $ ΗΠΑ)

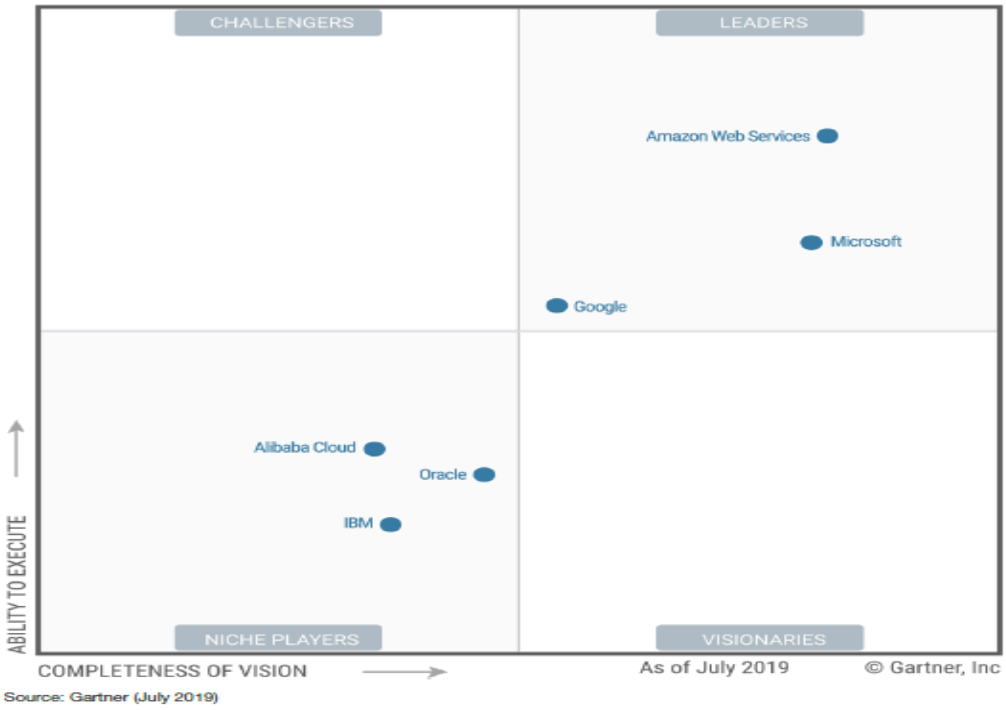
Προβλέψεις εσόδων για τις δημόσιες υπηρεσίες Cloud παγκοσμίως (δισ. $ ΗΠΑ)	**2018**	**2019**	**2020**	**2021**	**2022**
Cloud Business Process Services (BPaaS*)	45.8	49.3	53.1	57.0	61.1

Cloud Application Infrastructure Services (PaaS*)	15.6	19.0	23.0	27.5	31.8
Cloud Application Services (SaaS*)	80.0	94.8	110.5	126.7	143.7
Cloud Management and Security Services	10.5	12.2	14.1	16.0	17.9
Cloud System Infrastructure Services (IaaS*)	30.5	38.9	49.1	61.9	76.6
Total Market	**182.4**	**214.3**	**249.8**	**289.1**	**331.2**

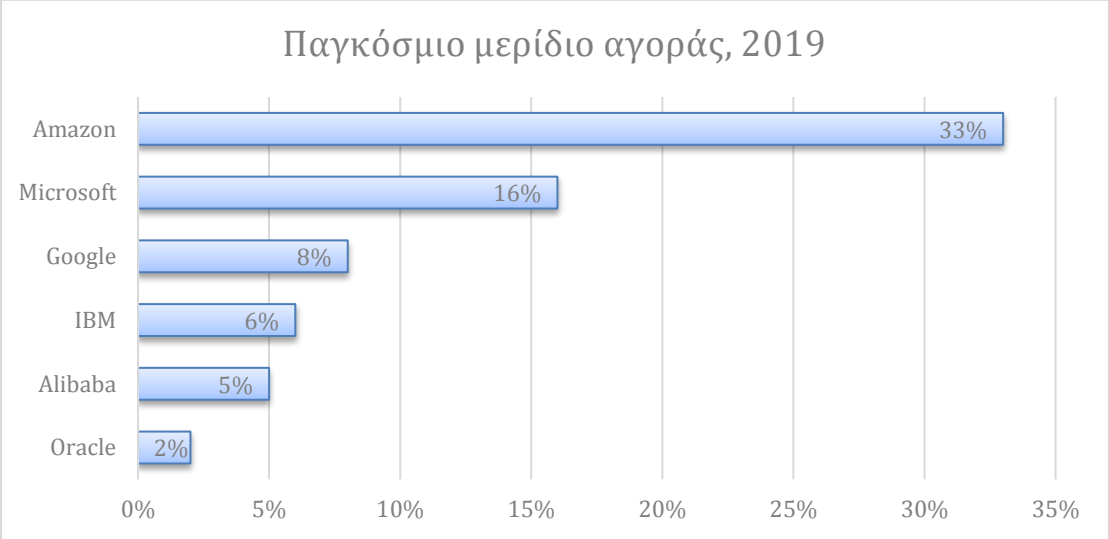
Πηγή: Gartner (April 2019)[3]
BPaaS = business process as a service; IaaS = infrastructure as a service; PaaS = platform as a service; SaaS = software as a service
Σημείωση: Τα σύνολα ενδέχεται να μην προστίθενται λόγω στρογγυλοποίησης.

Η αγορά των παρόχων υπηρεσιών Cloud παρουσιάζει υψηλή μονοπώληση, ειδικά σε ότι αφορά τις δημόσιες υπηρεσίες του Cloud, με το μεγαλύτερο μέρος της αγοράς να μοιράζεται ανάμεσα σε έξι εταιρίες. Παρακολουθώντας προσεκτικότερα αυτήν την ομάδα θα παρατηρήσουμε επιπλέον ότι υπάρχουν μέλη δυο ταχυτήτων. Όπως είναι εμφανές στην εικόνα 2 και στο διάγραμμα 1 ο ηγέτης της αγοράς είναι η εταιρία Amazon Web Services (AWS) με άμεσο ανταγωνιστή την Microsoft και στην τρίτη θέση ακολουθεί η Google. Το δεύτερο τμήμα της αγοράς περιλαμβάνει τους συγκριτικά μικρότερους παρόχους όπως είναι η IBM, το AliCloud και η Oracle.

[3] *https://www.gartner.com/en/newsroom/press-releases/2019-04-02-gartner-forecasts-worldwide-public-cloud-revenue-to-g*



Εικόνα 2. Βασικοί πάροχοι Cloud[4]



Διάγραμμα 1. Παγκόσμιο μερίδιο αγοράς
Πηγή: medium[5]

2.1.4. Ανάλυση ανταγωνισμού

Το Cloud Computing έχει εξελιχθεί σε ένα τεράστιο και περίπλοκο οικοσύστημα τεχνολογιών, προϊόντων και υπηρεσιών δημιουργώντας μια οικονομία πολλών

[4] *https://pages.awscloud.com/Gartner-Magic-Quadrant-for-Infrastructure-as-a-Service-Worldwide.html?trk=ar_carousel*
[5] https://medium.com/@jaychapel/alibaba-cloud-market-share-2 019-25d30bc096f7

δισεκατομμυρίων δολαρίων, όπου πολλοί πάροχοι ανταγωνίζονται για ένα συνεχώς αυξανόμενο μερίδιο αγοράς. Παρακάτω ακολουθεί μια συνοπτική επισκόπηση των βασικών παικτών στην αγορά.

Amazon Web Services (AWS)

Η Amazon παρέχει υπηρεσίες Cloud σε ιδιώτες και οργανισμούς από το 2006. Μέσω της διαδικτυακής πλατφόρμας Amazon Web Services προσφέρονται πολυάριθμες επιχειρηματικές λύσεις χρησιμοποιώντας ολοκληρωμένες υπηρεσίες ιστού καθώς και μια μεγάλη γκάμα υπηρεσιών IaaS και PaaS. Όντας από τους πρώτους παρόχους Cloud η πλατφόρμα AWS το 2019 περιλάμβανε 169 διαφορετικές υπηρεσίες[6]. Οι πιο δημοφιλείς από αυτές συνοψίζονται στον πίνακα 2.

Πίνακας 2. Οι πιο δημοφιλείς AWS υπηρεσίες.

AWS EC2	**Amazon RDS**	**Amazon S3**	**Amazon VPC**	**Amazon SNS**
Amazon Key Management Service	**Amazon Cloudwatch**	**Amazon Route 53**	**Amazon DynamoDB**	**Amazon Data Trasfer**

Πηγή: 2ndWatch[7]

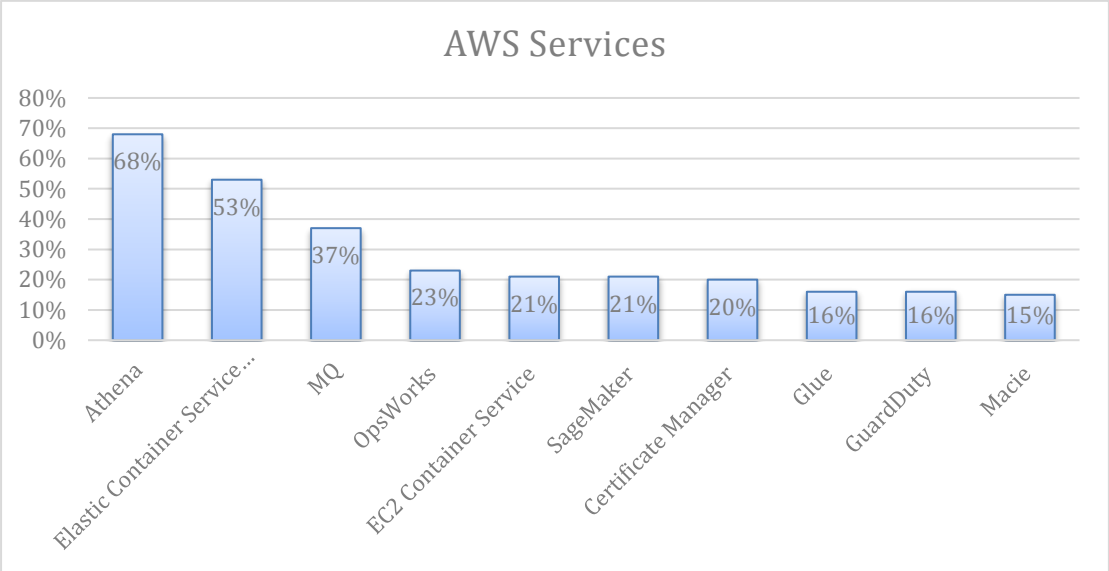
Επιπλέον η πλατφόρμα προσφέρει εργαλεία διαχείρισης, εποπτείας και ελέγχου των υπηρεσιών μέσω web client. Οι χρήστες έχουν πρόσβαση σε μια σειρά λειτουργιών, συμπεριλαμβανομένης της δημιουργίας και του ελέγχου ενός κλειδιού κρυπτογράφησης το οποίο και δύναται να χρησιμοποιηθεί για την ασφαλή αποθήκευση των δεδομένων τους.

Σε ότι αφορά το οικονομικό σκέλος των υπηρεσιών, η πλατφόρμα προφέρει τρεις διαφορετικούς τρόπους τιμολόγησης: ‘Pay as you Go’, ‘Save when you reserve’ και ‘Pay less using more’. Έτσι, οι πελάτες έχουν τη δυνατότητα να διαλέξουν ανάμεσα στα 3 μοντέλα και να καταλήξουν σε αυτό που ταιριάζει καλύτερα στις ανάγκες και επιθυμίες τους.

[6] https://medium.com/cloudpegboard/how-many-aws-services-are-there-51dda44fa946
[7] *https://www.2ndwatch.com/blog/popular-aws-products-2018/*

Ανταποκρινόμενη στο αυξανόμενο ενδιαφέρον για τα Big Data και την τεχνητή νοημοσύνη, η AWS συνεχώς αναπτύσσει νέες υπηρεσίες τεχνητής νοημοσύνης και εξόρυξης δεδομένων. Όπως φαίνεται και στο διάγραμμα 2 ανάμεσα στις υπηρεσίες με την μεγαλύτερη ανάπτυξη βρίσκονται και δύο (Athena και SageMaker) που ανήκουν στις προαναφερθείσες κατηγορίες υποδηλώνοντας την φιλοδοξία της πλατφόρμας να αποτελέσει σημείο αναφοράς για την ανάπτυξη των μελλοντικών εφαρμογών που θα βασίζονται σε πολλά δεδομένα και στο machine learning.

Τέλος, διαθέτοντας σχεδόν το ήμισυ της παγκόσμιας δημόσιας αγοράς υποδομών cloud, η Amazon είναι ο τωρινός ηγέτης της αγοράς. Το 2018, η Amazon ανακοίνωσε έσοδα ύψους 15,4 δισ. δολαρίων, σημειώνοντας αύξηση 26,8% σε σχέση με το προηγούμενο έτος. Για το 2019, η Amazon αναφέρει ότι τα έσοδα της από το πρώτο και δεύτερο τρίμηνο είναι 16,1 δισ. δολάρια, αύξηση 39% από το α 'εξάμηνο του 2018 και θεωρεί το 2019 ως επενδυτικό έτος, καθώς ενισχύει την τεχνολογική της ανάπτυξη και προσθέτει προσωπικό πωλήσεων.



Διάγραμμα 2. Οι ταχύτερα αναπτυσσόμενες AWS υπηρεσίες, 2018

Πηγή: 2nd Watch[8]

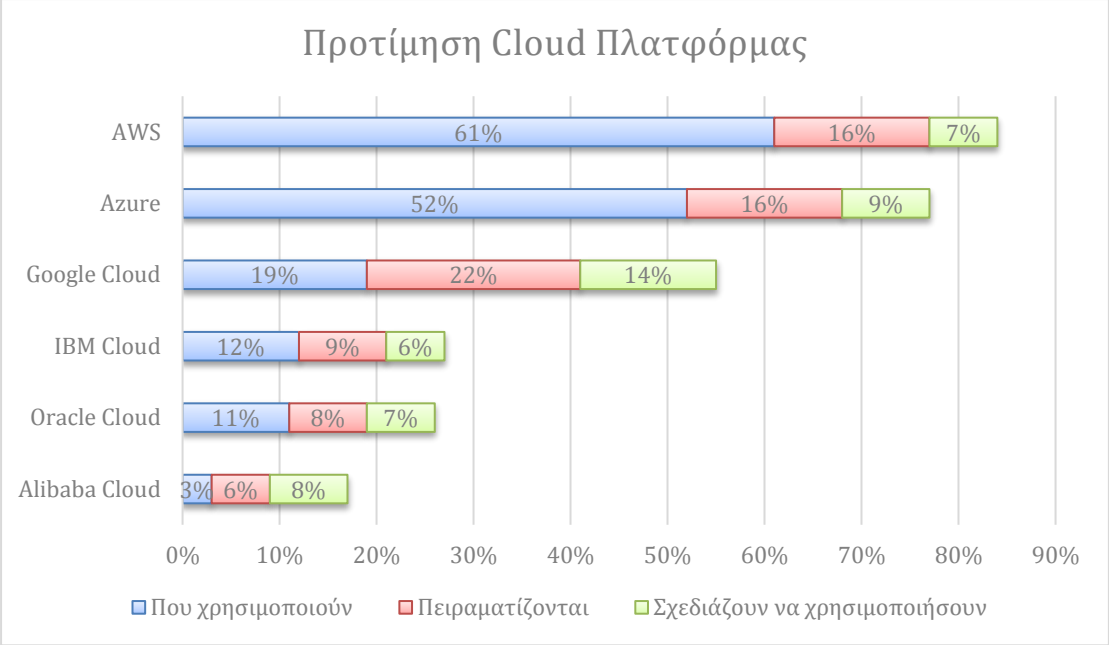
[8] https://www.zdnet.com/article/aws-big-data-machine-learning-services-gain-traction-says-2nd-watch/

Microsoft Azure

Η Microsoft Azure είναι η Cloud Computing λύση που προσφέρει η Microsoft. Με πρώτη κυκλοφορία το 2010, το Azure προσφέρει μια μεγάλη γκάμα από υπηρεσίες τύπου PaaS και SaaS και προβάλλοντας την εύκολη ενσωμάτωση των υπηρεσιών με τις υπάρχουσες υποδομές και λογισμικά Microsoft των πελατών της. Με την υπηρεσία RemoteApp για παράδειγμα παρέχεται η δυνατότητα εγκατάστασης και εκτέλεσης εφαρμογών σε Windows Server και παράδοσης τους εικονικά στον τελικό χρήστη χωρίς καμία εγκατάσταση σε φυσικό επίπεδο αλλά με την ίδια εμπειρία σαν η εφαρμογή να ήταν εγκατεστημένη τοπικά. Επίσης η πλατφόρμα ως άμεσος ανταγωνιστής της AWS επιδιώκει να προβάλλει το πλεονέκτημα που διαθέτει στην φιλοξενία Windows προϊόντων, όπως SQL Server και Windows Server, στην υποδομή του.

Η πλατφόρμα Microsoft Azure προσφέρει δωρεάν δωδεκάμηνη δοκιμή, η οποία περιλαμβάνει πρόσβαση σε όλες τις δημοφιλείς υπηρεσίες, πίστωση $ 200 (£ 153.74) και πάνω από 25 υπηρεσίες προσφέρονται πάντα δωρεάν. Η σελίδα περιλαμβάνει έναν υπολογιστή κόστους και μια υπηρεσία Pay as you go. Έτσι, κάθε σχέδιο μπορεί να προσαρμοστεί σε συγκεκριμένες ανάγκες.

Στο επιχειρηματικό κομμάτι, η ικανότητα της Microsoft να στοχεύει βιομηχανίες και startups είναι αδιαμφησβήτητη και εκμεταλλευόμενη αυτό το πλεονέκτημά της έχει καταφέρει να τις εξασφαλίσει σημαντικό μερίδιο αγοράς. Ειδικότερα, η Microsoft κέρδισε μεγάλους λιανοπωλητές που δεν επιθυμούν να συνεργαστούν με την AWS, καθώς ανταγωνίζονται με την Amazon.



Διάγραμμα 3. Προτίμηση Cloud πλατφόρμας

Πηή: ZDNet[9]

Τέλος, με μερίδιο αγοράς 15,5%, η Gartner εκτιμά τα ετήσια έσοδα της Azure στα 5 δισεκατομμύρια δολάρια. Η Microsoft αναμένεται να πάρει μερίδιο αγοράς από την AWS, αναφέροντας αύξηση της τάξης του 70% στα έσοδα Azure. Συνολικά, τα έσοδα από τις εμπορικές συναλλαγές του cloud της Microsoft το πρώτο και το δεύτερο τρίμηνο σημείωσαν 20,6 δισ. δολάρια, δηλαδή αύξηση κατά 40% στο α 'εξάμηνο του 2018.

Google Cloud Platform (GCP)

Η Google Cloud Platform είναι η σουίτα υπηρεσιών Cloud που προσφέρεται από την Google. Η πλατφόρμα επιτρέπει στους χρήστες να δημιουργούν επιχειρησιακές λύσεις προσφέροντας ένα ευρύ φάσμα υπηρεσιών, συμπεριλαμβανομένων των λύσεων IaaS και PaaS.

Οι βασικές υπηρεσίες περιλαμβάνουν το Compute Engine, App Engine, Container Engine, Cloud Storage και Big Query. Η Google προσφέρει επίσης ομαλή μετάβαση σε εικονικές μηχανές με ευέλικτη τιμολόγηση. Επιπλέον, υπάρχει δωρεάν δωδεκάμηνη δοκιμή, η οποία περιλαμβάνει $ 300 (£ 230,62) για όλες τις υπηρεσίες και τα προϊόντα

[9] https://www.zdnet.com/article/top-cloud-providers-2019-aws-microsoft-azure-google-cloud-ibm-makes-hybrid-move-salesforce-dominates-saas/

που προσφέρονται από την πλατφόρμα Google Cloud. Η Google ξεκίνησε ένα ενδιαφέρον σχέδιο τιμολόγησης για το cloud storage που θα μπορούσε να δώσει στις εταιρείες πιο προβλέψιμες δαπάνες.

Οι υπηρεσίες της πλατφόρμας είναι ιδιαίτερα συμβατές και υποστηρίζουν ανάπτυξη λύσεων που κάνουν χρήση τεχνολογιών container orchestration αλλά και την ανάπτυξη εφαρμογών σε Android. Στο Google Cloud Next, η εταιρεία δημιούργησε περισσότερους δεσμούς με νέους παίκτες στην αγορά του υβριδικού cloud μέσω μιας προσπάθειας που ονομάζεται Anthos διαμηνύοντας ότι αποτελεί την πρώτη πλατφόρμα για υβριδικές αλλά και multi-cloud λύσεις.

Alibaba

Η Alibaba είναι κορυφαίος πάροχος υπηρεσιών cloud στην Κίνα. Κατέχει ένα σημαντικό μερίδιο της αγοράς και ένα ολοκληρωμένο σύνολο υπηρεσιών μέσω των elastic computing, των υπηρεσιών βάσης δεδομένων, της δικτύωσης και των λύσεων CDN. Προσφέρει επίσης δωρεάν δοκιμή που καλύπτει 16 προϊόντα, ενώ διαφημίζει άλλα 20 ως "always free". Εκτός από αυτό, η εταιρεία δίνει μια πίστωση $ 300 (£ 230) για να δοκιμάσετε τις υπηρεσίες της.

Επιπλέον, κατέχει μερίδιο αγοράς 7,7%, σύμφωνα με την Gartner, με ετήσια έσοδα ύψους 2,49 δισ. δολαρίων και αύξηση 92,6% το 2018. Συνεχίζει την ανάπτυξη του το 2019, αναφέροντας συνολικά έσοδα 2,2 δισ. και αύξηση 66%. Αυτό τοποθετεί τα ετήσια έσοδα της εταιρείας πάνω από 4 δισεκατομμύρια δολάρια.

Η εταιρία έχει ένα ισχυρό πλεονέκτημα στην Κίνα, αλλά έχει και παγκόσμιες φιλοδοξίες, για αυτό και στοχεύει να επεκταθεί επενδύοντας στην ανάπτυξη του cloud. Μερικά παραδείγματα φαίνονται παρακάτω:

- Είναι αποκλειστικός προμηθευτής της Salesforce στην Κίνα.
- Άνοιξε το δεύτερο κέντρο δεδομένων στην Ιαπωνία.
- Ξεκίνησε μια εταιρική σχέση τεχνητής νοημοσύνης με την Intel.
- Επεκτάθηκε στην Πολωνία.
- Ανέπτυξε μια εταιρική σχέση υψηλού προφίλ με τη Διεθνή Ολυμπιακή Επιτροπή.

IBM

Το IBM Cloud είναι ένα σύνολο υπηρεσιών cloud computing που προσφέρει: πλατφόρμα ως υπηρεσία (PaaS), λογισμικό ως υπηρεσία (SaaS) και υποδομή ως υπηρεσία (IaaS). Οι παραπάνω υπηρεσίες δεν είναι όλες βασισμένες στο cloud, περιλαμβάνουν τόσο τους εικονικούς όσο και τους βασισμένους σε υλικό διακομιστές, δημόσια και ιδιωτικά δίκτυα όλα μέσω μιας cloud-based πλατφόρμας, παρέχοντας πλήρη έλεγχο της υποδομής.

Η εταιρία δίνει την δυνατότητα στο χρήστη να προσαρμόσει το πακέτο των υπηρεσιών στις ανάγκες του, ώστε να πληρώσει μόνο για αυτές που θα χρησιμοποιήσει. Επιπρόσθετα, ο χρήστης έχει τη δυνατότητα χρήσης ενός «Lite» επιπέδου με 256MB Cloud Foundry Memory, χωρίς χρονικό όριο και χωρίς να ζητηθούν τα στοιχεία της πιστωτικής κάρτας. Επιπλέον, η εταιρία δίνει τη δυνατότητα του υπολογισμού του κόστους των λειτουργιών της.

Έχοντας πολλά κοινά, η στρατηγική της IBM και η προσέγγισή της για το AI, είχε ως αποτέλεσμα το 2018, η εταιρία να εγκαινιάσει το OpenScale για AI, το οποίο σχεδιάστηκε για τη διαχείριση πολλαπλών εργαλείων AI.

Τέλος, σύμφωνα με την Gartner, η IBM κατέχει μερίδιο αγοράς 1,8% με ετήσια έσοδα 577 εκατομμύρια δολάρια και αύξηση 24% το 2018.

Oracle Cloud

Το Oracle Cloud είναι άλλη μια υπηρεσία cloud computing που προσφέρεται από την Oracle Corporation. Μια πρόσφατη ανασκόπηση του Forrester επισήμανε ότι οι υπηρεσίες της Oracle υποστήριζαν ένα μεγάλο εύρος διαφορετικών εργασιών, ειδικότερα για IoT, OLTP, microservices, μαζί με εφαρμογές που εξαρτώνται από την AI και τη μηχανική μάθηση. Υπάρχουν δύο βασικές διατάξεις παροχής υπηρεσιών: αρχιτεκτονική cloud και δεδομένα αποθήκευσης (storage data).

Η αρχιτεκτονική cloud περιλαμβάνει διαχείριση δεδομένων, βάσεις δεδομένων και εφαρμογές, ενώ το Oracle Data Cloud προωθεί αναλυτικά στοιχεία μεγάλου όγκου για πληροφορίες επιχειρηματικής ευφυΐας. Η Oracle προσφέρει επίσης μια σειρά από πλατφόρμες SaaS (Λογισμικό ως Υπηρεσία) όπως τα εργαλεία HCM, EPM, SCM και

κοινωνικά μέσα. Η Oracle παρέχει επίσης ένα ευρύ φάσμα υπηρεσιών cloud, που ικανοποιούν ανάγκες μεγάλων επιχειρήσεων.

Για όσους κάνουν εγγραφή εκτός από μια δωρεάν δοκιμαστική περίοδο 30 ημερών, το Oracle Cloud προσφέρει επίσης μια δωρεάν βαθμίδα υπηρεσιών η οποία περιλαμβάνει απεριόριστη πρόσβαση σε δύο αυτόνομες βάσεις δεδομένων που συνοδεύουν το Oracle Application Express (APEX) και την Oracle SQL.

2.1.5. Πλεονεκτήματα και μειονεκτήματα

Έπειτα από τον ορισμό του Cloud Computing και την συνοπτική επισκόπηση της αγοράς, θα προσπαθήσουμε να απαντήσουμε στο ερώτημα εάν η ανάπτυξη μιας σημερινής εφαρμογής θα πρέπει να γίνει χρησιμοποιώντας τεχνολογίες του Cloud ή τις πιο "παραδοσιακές" τεχνικές ανάπτυξης offline, on-premise, επιτραπέζιων (desktop) εφαρμογών. Η προσπάθεια μας αυτή θα ξεκινήσει με την απαρίθμηση των βασικών πλεονεκτημάτων που παρέχει το Cloud καθώς και αναφορά στα μειονεκτήματα έναντι των παραδοσιακών τεχνικών. Σε αυτό το σημείο να σημειώσουμε ότι η παράθεση των στοιχείων γίνεται έχοντας συμβουλευτεί άρθρα μεγάλων παρόχων του Cloud καθώς επίσης και την σχετική βιβλιογραφία. Τα οφέλη χρήσης του Cloud όπως θα δούμε έχουν τόσο τεχνικές όσο και οικονομικές ή επιχειρησιακές διαστάσεις και αρκετές φορές η κατηγοριοποίηση τους δεν είναι εύκολη.

Ένα από τα πολύ συχνά αναφερόμενα πλεονεκτήματα το οποίο προβάλλεται αρκετά συχνά τόσο από τους παρόχους όσο και από την βιβλιογραφία σχετίζεται με ένα από τα βασικά χαρακτηριστικά του Cloud, αυτό της ελαστικότητας (Ahmad Dar, 2018; Xue & Xin, 2016). Η δυνατότητα της κατά βούληση αυξομείωσης της υπολογιστικής δύναμης και των πόρων αποτελεί ένα από τα πιο ελκυστικά χαρακτηριστικά των υπηρεσιών του Cloud προσδίδοντας στις επιχειρήσεις και οργανισμούς που τις χρησιμοποιούν μεγάλη ευελιξία. Η αυξημένη ευελιξία όμως δεν αναφέρεται μόνο την τεχνική πλευρά της αυξομείωσης των χρησιμοποιούμενων υποδομών, αλλά και σε αυτήν που απολαμβάνουν πελάτες και εργαζόμενοι ενός οργανισμού χρησιμοποιώντας υπηρεσίες βασιζόμενες στο Cloud. Μεταξύ άλλων χρήση των υπηρεσιών ανεξαρτήτως τοποθεσίας όπως και η παύση αυτών οποιαδήποτε στιγμή χωρίς επιπλέον κόστος (; Xue & Xin, 2016).

Το δεύτερο πλεονέκτημα είναι αυτό που σχετίζεται με τα οικονομικά οφέλη της χρήσης του Cloud (Ahmad Dar, 2018; Xue & Xin, 2016; Singh, 2019). Αυτά συνήθως επεκτείνονται σε τρία διακριτά στοιχεία: 1) μείωση κεφαλαιακών δαπανών για την αγορά υπολογιστικής υποδομής 2) μείωση μεταβλητών δαπανών για τη λειτουργία και συντήρηση της υπολογιστικής υποδομής 3) αποδοτικότερη σχέση μεταβλητών δαπανών για την προμήθεια υπολογιστικών πόρων καθώς οι τελευταίοι παρέχονται στα πλαίσια μια οικονομίας κλίμακας. Με μια πιο οικονομικά ευρεία ματιά θα μπορούσαμε να πούμε ότι η επιλογή του Cloud μας επιτρέπει μεγαλύτερη οικονομική ευελιξία και μικρότερη έκθεση στον κίνδυνο εξ' αιτίας του μοντέλου πληρωμή-με-τη-χρήση.

Η ασφάλεια και η ακεραιότητα των δεδομένων αλλά και των υπολογισμών προσελκύει την προσοχή ολοένα και περισσότερων ειδικών, επιχειρήσεων, καταναλωτών καθώς αυξάνεται η προβολή των φυσικών δραστηριοτήτων στον ψηφιακό κόσμο και τα κρούσματα παραβίασης είναι σχεδόν καθημερινό φαινόμενο. Αν και κατά πολλούς η πτυχή της ασφάλειας είναι ένα από τα μειονεκτήματα του Cloud υπάρχει και η αντίθετη άποψη η οποία αναφέρει ότι δεδομένα και υπολογισμοί στο Cloud προστατεύονται από συγκεκριμένα πρωτόκολλα αποδεκτά για το επίπεδο τους από όλους τους αρμόδιους φορείς. Τα τελευταία εκτός του ότι προστατεύουν μια εφαρμογή σε ένα ευρύ φάσμα (δίκτυο, αποθήκευση δεδομένων, φυσική προστασία υλικού, εκτέλεση υπολογισμών) προσφέρουν εκτεταμένη εποπτεία και παρακολούθηση όλων των σχετικών συμβάντων δυνατότητα που επιτρέπει την αποτροπή κακόβουλων δράσεων ή τον εντοπισμό τέτοιων σε σύντομο χρονικό διάστημα. Το τελευταίο είναι ιδιαίτερα σημαντικό εάν αναλογιστούμε ότι η πιο πρόσφατη νομοθεσία που θεσπίστηκε στην Ε.Ε. σχετικά με την προστασία των προσωπικών δεδομένων ορίζει ότι σε περίπτωση απώλειας / κλοπής δεδομένων η αναφορά στον αρμόδιο φορέα πρέπει να πραγματοποιηθεί εντός 72 ωρών από το συμβάν[10].

Ένα συναφές χαρακτηριστικό με το προηγούμενο για το οποίο το Cloud έχει σαφές πλεονέκτημα είναι αυτό της αξιοπιστίας – διαθεσιμότητας μιας υπηρεσίας όπως επίσης και η επαναφορά από κάποιο καταστροφικό συμβάν. Καθώς οι υπηρεσίες βασισμένες στο Cloud εκτελούνται από πόρους οι οποίοι είναι εικονικοί και το υλικό στο οποίο βασίζονται γεωγραφικά κατανεμημένο, τυχόν σφάλματα που προκύψουν τόσο σε

[10] https://gdpr-info.eu/art-33-gdpr/

επίπεδο λογισμικού όσο και σε επίπεδο υλικού από οποιαδήποτε αιτία (αστοχία υλικού, φυσική καταστροφή) αντικαθίστανται άμεσα από κάποιο άλλο διαθέσιμο.

Βασισμένο πάλι στις τεχνολογίες εικονικοποίησης (virtualization) και αυτοματισμού το Cloud έχει την δυνατότητα να προσφέρει μια σειρά από διευκολύνσεις και λύσεις για την ελάφρυνση του φόρτου λειτουργίας και συντήρησης της υποδομής μιας διαδικτυακής εχφαρμογής. Ειδικότερα, είναι στις αρμοδιότητες του παρόχου να συντηρεί το υλικό και λογισμικό (μέχρι το σημείο που προβλέπει το SLA – εξαρτάται από την παρεχόμενη υπηρεσία) με τις απαραίτητες ενημερώσεις επιτρέποντας στον οργανισμό / επιχείρηση να διαθέσει περισσότερους πόρους στην επιχειρηματική λογική και να αποδεσμεύσει αντίστοιχα από διαδικασίες που δεν προσφέρουν άμεσα στις δραστηριότητές της μειώνοντας το κόστος των προϊόντων / υπηρεσιών της ενώ συνάμα αυξάνει την ποιότητα τους όπως και την ταχύτητα κυκλοφορίας στην αγορά (Ahmad Dar, 2018), (Xue & Xin, 2016), (Singh, 2019).

Στον αντίποδα, ένας από τα βασικότερους λόγους ανάσχεσης της τάσης υιοθέτησης του Cloud είναι η ιδιωτικότητα των δεδομένων (Ahmad Dar, 2018). Καθώς τα δεδομένα δεν βρίσκονται υπό τον έλεγχο του ιδιοκτήτη, αρκετές επιχειρήσεις αποκλείουν το δημόσιο Cloud ή επιλέγουν λύσεις ιδιωτικού Cloud με ιδιόκτητες εγκαταστάσεις για τις υποδομές που φιλοξενούν τα δεδομένα τους. Σε πολλές περιπτώσεις η μεταφορά τέτοιων δεδομένων εκτός κάποιας γεωγραφικής περιοχής είναι απαγορευμένη από το νόμο κάνοντας έτσι την λύση του Cloud λιγότερη ιδανική.

Ένας ακόμα ανασταλτικός παράγοντας που αναφέρουν οι επιχειρήσεις είναι ο κίνδυνος που διατρέχουν να εγκλωβιστούν σε ένα πάροχο (vendor lock-in) (Ahmad Dar, 2018). Λόγω έλλειψης κλαδικών προτύπων και ομοιογένειας υπηρεσιών η μετακίνηση σε άλλο πάροχο από τον αρχικό ίσως αποβεί δύσκολη και κοστοβόρα διαδικασία.

Τέλος ένα προφανές μειονέκτημα της τεχνολογίας του Cloud είναι η αδυναμία λειτουργίας των υπηρεσιών / εφαρμογών όταν δεν υπάρχει συνδεσιμότητα στο διαδίκτυο.

Φυσικά τα παραπάνω πλεονεκτήματα αλλά και μειονεκτήματα δεν έχουν καθολική εφαρμογή υπό την έννοια ότι το Cloud ίσως αποδειχτεί πιο επωφελές για κάποια είδη και μεγέθη οργανισμών ή ακόμα και για διαφορετικούς κλάδους. Ενδεικτικά αναφέρουμε την έρευνα των Chen, Ta-Tao, & Kazuo, (2016) η οποία επισημαίνει ότι τα οφέλη του Cloud είναι υψηλότερα σε υποστηρικτικές διαδικασίες παρότι σε

πρωταρχικές, οι μεγαλύτεροι οργανισμοί που έκαναν χρήση των SaaS λύσεων είδαν μεγαλύτερες εξοικονομήσεις από ότι οι μικρομεσαίες επιχειρήσεις και η πολύ μικρές επιχειρήσεις βοηθήθηκαν στην επεκτασιμότητα των δραστηριοτήτων τους από την χρήση λύσεων της κατηγορίας PaaS συγκριτικά με μικρές και μεσαίες.

2.1.6. Υπολογιστική άνευ διακομιστών (Serverless Computing)

Μια από τις πιο πρόσφατες τεχνολογικές καινοτομίες στον τομέα του Cloud Computing είναι η Υπολογιστική άνευ Διακομιστών (Serverless Computing). Η τελευταία αναφέρεται σε αρχιτεκτονικές και πρακτικές ανάπτυξης υπηρεσιών και συστημάτων χωρίς να απαιτείται η συμπερίληψη στον σχεδιασμό ή / και στην υλοποίηση τεχνικών λεπτομερειών σχετικά με την εγκατάσταση και χρήση των διακομιστών που θα τις φιλοξενήσουν. Οι Castro, Ishakian, Muthusamy, & Slominski, (2019) ερμηνεύουν τον όρο ως την υπολογιστική πλατφόρμα η οποία αποκρύπτει την χρήση διακομιστών από τους προγραμματιστές και επιτρέπει την εκτέλεση κώδικα κατά βούληση, είναι αυτόματα επεκτάσιμη και η χρέωση γίνεται μόνο κατά την ώρα της εκτέλεσης του κώδικα. Από τον ορισμό εύκολα εξάγονται και τα κυρίαρχα χαρακτηριστικά της τεχνολογίας:

1. Κόστος ανά χρήση (pay-as-you-go) – απόκρυψη των διακομιστών σημαίνει επίσης και των συνεπαγόμενων χρεώσεων για την λειτουργία τους. Έτσι η χρέωση βαραίνει μόνο την εκτέλεση του κώδικα και φυσικά μόνο ενόσω αυτή λαμβάνει χώρα.
2. Ελαστικότητα – απόρροια της αφαίρεσης των διακομιστών είναι και η δυνατότητα αύξησης της δυναμικότητας κατά βούληση όπως και την απόλυτη διακοπή εκτέλεσης του κώδικα (μηδενική δυναμικότητα) χωρίς δυσκολία.

Νέες αρχιτεκτονικές πρακτικές, κυριότερα αυτή των μικροϋπηρεσιών (microservices) (Soldani, Tamburri, & Van Den Heuvel, 2018), η μεγάλη πρόοδος των τεχνολογιών της εικονικοποίησης (virtualization), με πιο μεγάλη αυτή των κοντέινερ (containerization), οδήγησαν στην στροφή των προγραμματιστών και των αρχιτεκτόνων των διαδικτυακών υπηρεσιών προς την σχεδίαση και ανάπτυξη πιο μικρών, αυτόνομων υπηρεσιών οι οποίες δεν διατηρούν κατάσταση (stateless) (Pérez, Moltó, Caballer, & Calatrava, 2018). Από την πλευρά των παρόχων Cloud οι εξελίξεις αυτές τους οδήγησαν στην σχεδίαση και προσφορά υπηρεσιών οι οποίες υποστηρίζουν

την προαναφερθείσα αρχιτεκτονική στροφή οδηγώντας έτσι στην ανάδυση μεγάλων Serverless οικοσυστημάτων. Όπως η Amazon περιγράφει στην ιστοσελίδα της για την τεχνολογία Serverless: είναι η φυσική αρχιτεκτονική του Cloud η οποία επιτρέπει την μετατόπιση λειτουργικών ευθυνών στον πάροχο (υπηρεσιών Cloud) με στόχο την αυξημένη ευελιξία και καινοτομία.

Παρότι ο όρος Serverless αναφέρεται σε ένα αρκετά ευρύ φάσμα υπηρεσιών με τα παραπάνω χαρακτηριστικά η πιο συνηθισμένη μορφή είναι αυτή των FaaS (Function as a Service) η οποία, όπως προδίδει και το όνομα της, επιτρέπει την εκτέλεση κώδικα σε απόκριση διαφόρων συμβάντων ή HTTP αιτημάτων. Υπάρχουν πλέον πολυάριθμες υλοποιήσεις FaaS τόσο ανοικτού κώδικα (OpenWhisk, OpenFaaS, IronFunctions) όσο και υλοποιήσεις-πλατφόρμες κλειστού κώδικα που προσφέρονται από τους μεγάλους παρόχους (AWS Lambda, Google Functions, Azure Functions). Οι υλοποιήσεις αυτές, οι οποίες ξεκίνησαν το 2014 (AWS Lambda) αρχικά προσέφεραν την εκτέλεση μικρών τμημάτων κώδικα για ένα μικρό χρονικό διάστημα αλλά πολύ γρήγορα τα χαρακτηριστικά τους και η αξιοπιστία τους αναπτύχθηκε στον βαθμό που, σε συνδυασμό με άλλες υπηρεσίες τους Serverless οικοσυστήματος, να αποτελούν την ραχοκοκαλιά ολόκληρων εφαρμογών.

Στην βιβλιογραφία αλλά και στον κλάδο η τεχνολογία Serverless αποτιμάται ως την εξέλιξη του μοντέλου PaaS του Cloud Computing η οποία υπόσχεται να προσφέρει ότι αρχικά είχε υποσχεθεί το Cloud και σε μεγάλο βαθμό το επιτυγχάνει. Ειδικότερα η τεχνολογία εκτός του προφανούς οικονομικού πλεονεκτήματος που προσφέρει, αναφερόμαστε στην δυνατότητα να συσχετίζονται τα κόστη σταθερού κεφαλαίου με την χρήση του, είναι επίσης εύκολη και απλή στην χρήση και δεν προϋποθέτει μεγάλο αριθμό προγραμματιστών ή διαχειριστών.

Φυσικά όπως κάθε νέα τεχνολογία έτσι και στην περίπτωση της Υπολογιστικής άνευ Διακομιστών υπάρχουν αρκετά εμπόδια και προκλήσεις στον δρόμο για την υιοθέτηση της από το ευρύ καταναλωτικό κοινό του Cloud. Σε τεχνικό επίπεδο η αδυναμία των πλατφορμών FaaS να υποστηρίξουν όλες τις γλώσσες προγραμματισμού αποτελεί έναν ανασταλτικό παράγοντα για όσους σκέφτονται Serverless εναλλακτικές όπως επίσης και ότι η τεχνολογία δεν ενδείκνυται για συστήματα στα οποία η απόκριση (σε επίπεδο δευτερολέπτου) είναι καθοριστικής σημασίας. Προβληματισμός όμως απορρέει και από το γεγονός ότι τέτοιου είδους υπηρεσίες επισύρουν τον κίνδυνο εγκλωβισμού σε έναν πάροχο (vendor lock-in) (Adzic & Chatley, 2017).

Παρόλα αυτά η ανάπτυξη άνευ Διακομιστών εφαρμογών, υπηρεσιών και λύσεων τα τελευταία χρόνια είναι αδιαμφισβήτητη και ακόμα περισσότερο η έρευνα και οι πειραματισμοί με αυτήν έχουν εκτοξευθεί σε μια απόπειρα τόσο της ακαδημαϊκής όσο και της επαγγελματικής κοινότητας να διαπιστώσει τα όρια της εν λόγω τεχνολογίας (Yussupov, Breitenbücher, Leymann, & Wurster, 2019).

2.2. Γεωγραφικά Πληροφοριακά Συστήματα - GIS

Σε αυτό το κεφάλαιο θα ορίσουμε την έννοια των Γεωγραφικών Πληροφοριακών Συστημάτων – GIS, θα αναλύσουμε την εξέλιξη τους στο χρόνο και τις σχετικές τεχνολογίες στο Cloud.

2.2.1. Ορισμός

Γεωγραφικό Πληροφοριακό Σύστημα είναι ένα πλαίσιο συλλογής, διαχείρισης και ανάλυσης πολλών τύπων δεδομένων, οι ρίζες του οποίου βρίσκονται στην επιστήμη της Γεωγραφίας. Αναλύει χωρικά δεδομένα και οργανώνει τα διαφορετικά επίπεδα (layers) της πληροφορίας σε οπτικοποιήσεις χρησιμοποιώντας χάρτες. Χρησιμοποιώντας αυτήν την τεχνική ένα ΓΠΣ αποκαλύπτει βαθύτερες πληροφορίες σε συλλογές δεδομένων όπως μοτίβα, σχέσεις και καταστάσεις[11]. Άλλοι ορισμοί δίνουν περισσότερη έμφαση στην υποδομή ενός τέτοιου συστήματος οπότε και κάνουν λόγο για ένα υπολογιστικό σύστημα για την συλλογή, αποθήκευση, διαχείριση, ανάλυση και απεικόνιση δεδομένων και τις σχετικές τους θέσεις πάνω στην επιφάνεια της Γης.

2.2.2. Ιστορικά στοιχεία

Η σημασία της χωρικής διάστασης των δεδομένων και κατ' επέκταση η ανάγκη για ένα εργαλείο καταγραφής και ανάλυσης της εμφανίστηκε για πρώτη φορά το 19ο αιώνα όταν επιστήμονες την εποχής όπως ο Charles Picquet και ο John Snow χρησιμοποίησαν τεχνικές γεωχωρικής ανάλυσης για την διερεύνηση μεταδοτικών μοτίβων μολυσματικών ασθενειών. Συγκεκριμένα, ο επιδημιολόγος John Snow το 1854 διαπίστωσε ότι μια επιδημία χολέρας σχετιζόταν με μια μολυσμένη πηγή νερού στο Σόχο του Λονδίνου και χρησιμοποίησε τεχνικές GIS (layering) για να απεικονίσει την

[11] https://www.esri.com

χωρική συσχέτιση των κρουσμάτων με την πηγή. Η εμφάνιση των GIS με την σημερινή τους μορφή, δηλαδή ως μια πληροφοριακή τεχνολογία με τον υπολογιστή να αποτελεί το μέσο για την διαχείριση και ανάλυση των δεδομένων, χρονολογείται το 1960 μαζί με την ανάδυση της ποσοτικής και υπολογιστικής γεωγραφίας. Το πρώτο GIS αποδίδεται στον Roger Tomlinson για την ανάπτυξη του πρώτου ΓΠΣ βασισμένο στου υπολογιστές το 1963 για την κυβέρνηση του Καναδά. Ο Tomlinson οραματιζόταν την χρήση ηλεκτρονικών υπολογιστών ώστε να δημιουργήσει ένα αποθετήριο όπου θα συγκέντρωνε δεδομένα για τους πόρους από όλες τις περιοχές του Καναδά. (Wieczorek & Delmerico, 2009)

Μέχρι τα μέσα της δεκαετίας του 90’ η αγορά των ΓΠΣ μονοπωλούνταν από το “κλειστά” (proprietary) λογισμικά και πρότυπα ενώ η μέθοδος αποθήκευσης των πληροφοριών ήταν τα αρχεία. Η προηγούμενη κατάσταση είχε σχεδιαστεί στον άξονα της ταχείας ανάγνωσης δεδομένων από λογισμικά του ίδιου τύπου. Όπως εύκολα μπορεί να αντιληφθεί κανείς το σύστημα αυτό καθιστούσε πολύ δύσκολο τον διαμοιρασμό των πληροφοριών ειδικά ανάμεσα σε οργανισμούς που χρησιμοποιούσαν διαφορετικά λογισμικά ΓΠΣ. Η πρώτη απάντηση σε αυτό το εμπόδιο δόθηκε με την ανάδυση των γεω-σχεσιακών μοντέλων. Τα τελευταία αποθήκευαν περιγραφικές πληροφορίες σε μια σχεσιακή βάση δεδομένων και συνδέονταν με την γεωχωρική πληροφορία με την συσχέτιση τους με ένα αρχείο. Βασικό μειονέκτημα αυτής της λύσης ήταν η περιορισμένη επεκτασιμότητα της και το γεγονός ότι εξαιτίας της δυϊκής προσέγγισης στην αποθήκευση πληροφοριών για ίδιες οντότητες συνεπαγόταν επιπλέον φόρτο για την αποθήκευση και τήρηση των δεδομένων.

Στα μέσα της δεκαετίας του 90’ η τεχνολογία των βάσεων δεδομένων ικανές να αποθηκεύσουν αυτούσια την γεωχωρική πληροφορία κάνει την εμφάνιση της και δίνει λύση στην αποθήκευση των δεδομένων αλλά και επιτρέπει την χρήση της ισχύος της βάσης για την εκτέλεση γεωχωρικών ερωτημάτων.

2.2.3. Web & Cloud GIS

Οι τεχνολογικές εξελίξεις στον τομέα των υπολογιστικών συσκευών αλλά και των δικτύων από την δεκαετία του 90’ και αργότερα οδήγησαν στην χρήση GIS σε ένα αρκετά ευρύ φάσμα εφαρμογών. Μαζί με την εμφάνιση εφαρμογών ιστού έκαναν την εμφάνιση τους και τα πρώτα κατανεμημένα συστήματα GIS τα οποία στηρίζονταν στο

μοντέλο ενός GIS-server και ενός φυλλομετρητή σε ρόλο εφαρμογής πελάτη και χρησιμοποιούνταν για σχετικά απλές διαδικασίες όπως δρομολόγηση και χαρτογράφηση. Με άλλα λόγια τα πρώτα Web GIS αποτελούνταν από έναν «ισχυρό» GIS διακομιστή στον οποίο πραγματοποιούνταν όλες οι γεωχωρικοί υπολογισμοί, αυτοί ενσωματώνονταν σε μια σελίδα η οποία εν τέλει αποστέλλονταν στον πελάτη για εμφάνιση επιτυγχάνοντας έτσι για πρώτη φορά τον γεωγραφικό διαχωρισμό των γεωχωρικών πληροφοριών από τον καταναλωτή τους.[12]

Οι μεταγενέστερες εξελίξεις στις τεχνολογίες ιστού αλλά και η ανάπτυξη του τελευταίου σε ένα τεράστιο οικοσύστημα δεδομένων και υπηρεσιών γρήγορα οδήγησαν στην εξέλιξη των Web GIS από το μοντέλο ενός πελάτη-φυλλομετρητή και ενός διακομιστή σε ένα σύστημα διαφορετικών πελατών οι οποίοι αφενός επικοινωνούν με περισσότερα του ενός απομακρυσμένα συστήματα προκειμένου να αντλήσουν τις πληροφορίες που απαιτούνται και αφετέρου είναι σε θέση να συνθέσουν τα ανακτηθέντα δεδομένα και να εκτελέσουν πιο σύνθετες επεξεργασίες σε αυτά. Υπό μια έννοια έχουμε η περεταίρω κατανομή των στοιχείων ενός Web GIS βασιζόμενο σε μια υπηρεσιοστραφή αρχιτεκτονική (Service Oriented Architecture).

Όπως είναι εύκολα αντιληπτό βασική προϋπόθεση για την ακμή των Web GIS στην παραπάνω μορφή ήταν η καθιέρωση προτύπων επικοινωνίας και μεταφοράς των δεδομένων ώστε να εξασφαλιστεί η διαλειτουργικότητα των εμπλεκόμενων συστημάτων. Εξέλιξη σταθμός στην δημιουργία προτύπων ήταν η ίδρυση του OGC (Open Geospatial Consortium) το 1994, μια διεθνή κοινοπραξία η οποία έχει στόχο την ανάπτυξη ανοικτών προτύπων για την διαλειτουργικότητα των υπηρεσιών βασισμένων σε γεωχωρικές πληροφορίες. Τα πιο γνωστά πρότυπα που καθιερώθηκαν, γνωστά ως OpenGIS πρότυπα, είναι τα ακόλουθα:

- **Web Map Service (WMS):** Προσδιορίζει μια απλή HTTP διεπαφή για τα αιτήματα εικόνων με γεωαναφορά από μια ή περισσότερες γεωχωρικές βάσεις δεδομένων[13].
- **Web Processing Service (WPS):** Παρέχει κανόνες για την προτυποποίηση της εισόδου και εξόδου αναφορικά με τις υπηρεσίες γεωχωρικών επεξεργασιών

[12] https://esripress.esri.com/storage/esripress/images/188/115391_webgis_chapter01.pdf
[13] https://www.opengeospatial.org/standards/wms

όπως η επικάλυψη πολυγώνων. Επίσης προσδιορίζει μια διεπαφή για την δημοσίευση των γεωχωρικών επεξεργασιών και την εύρεση τους από πελάτες[14].

- **Web Feature Service (WFS):** Προσδιορίζει τον τρόπο με τον οποίο η γεωγραφική πληροφορία δημιουργείται και τροποποιείται. Ειδικότερα προσδιορίζει ενέργειες αναζήτησης, ερωτημάτων, κλειδώματος, συναλλαγής αλλά και ενέργειες διαχείρισης αποθηκευμένων και παραμετροποιημένων εκφράσεων ερωτημάτων[15].
- **GML & GeoJSON:** Η GML αποτελεί μια XML γραμματική ώστε τα γεωχωρικά αντικείμενα να μπορούν να εκφραστούν σε XML. Έχει τον ρόλο γλώσσας μοντελοποίησης των γεωγραφικών συστημάτων αλλά και τον ρόλο ανοικτού τύπου ανταλλαγής για τις διάφορες γεωχωρικές συναλλαγές μέσω διαδικτύου[16]. Παρόμοια λειτουργία με τον τύπο GML έχει και ο τύπος GeoJSON αν και με μικρότερες δυνατότητες. Ο τελευταίος κωδικοποιεί συλλογές γεωγραφικών αντικειμένων μαζί με περιγραφικά δεδομένα χρησιμοποιώντας την μορφή JSON. Τα GeoJSON αντικείμενα μπορούν να αναπαριστούν μια γεωμετρία, ένα αντικείμενο ή μια συλλογή αντικειμένων[17].

Με την ανάπτυξη του διαδικτύου και του παγκόσμιου ιστού άρθηκε εμπόδιο της απόστασης από τον ψηφιακό κόσμο επιτρέποντας έτσι άμεση πρόσβαση σε πληροφορίες ανεξαρτήτως γεωγραφικής θέσης. Αυτό το χαρακτηριστικό αποτελεί και την βασική πηγή πλεονεκτημάτων των Web-GIS σε αντιπαραβολή των παραδοσιακών επιτραπέζιων GIS. Παρακάτω αναφέρουμε μερικά από τα απορρέοντα πλεονεκτήματα:

- Παγκόσμια πρόσβαση -
- Μεγάλος αριθμός χρηστών
- Υποστήριξη μεγάλου αριθμού πλατφορμών
- Φιλικά στον οποιοδήποτε χρήστη
- Χαμηλότερο ανα χρήστη κόστος
- Ποικιλία εφαρμογών

[14] https://www.opengeospatial.org/standards/wps
[15] https://www.opengeospatial.org/standards/wfs
[16] https://www.opengeospatial.org/standards/gml
[17] https://www.opengeospatial.org/standards/eo-geojson

3. Σχεδίαση Συστήματος & Εργαλεία Ανάπτυξης

Το παρόν κεφάλαιο ουσιαστικά αποτελεί την δεύτερη ενότητα της διπλωματικής εργασίας. Διότι από εδώ και στο εξής θα παρουσιαστούν διαδοχικά α) η δομή β) τα εργαλεία ανάπτυξης. Θα κάνουμε την περιγραφή της αρχιτεκτονικής του πρωτοτύπου στα πλαίσια της οποίας θα παρουσιαστούν και θα αναλυθούν τα συστατικά στοιχεία που εισάγει η προσέγγιση "άνευ διακομιστών". Ακολούθως θα παρουσιάσουμε τα εργαλεία που επιλέχθηκαν για την υλοποίηση της παραθέτοντας τους λόγους για την επιλογή τους.

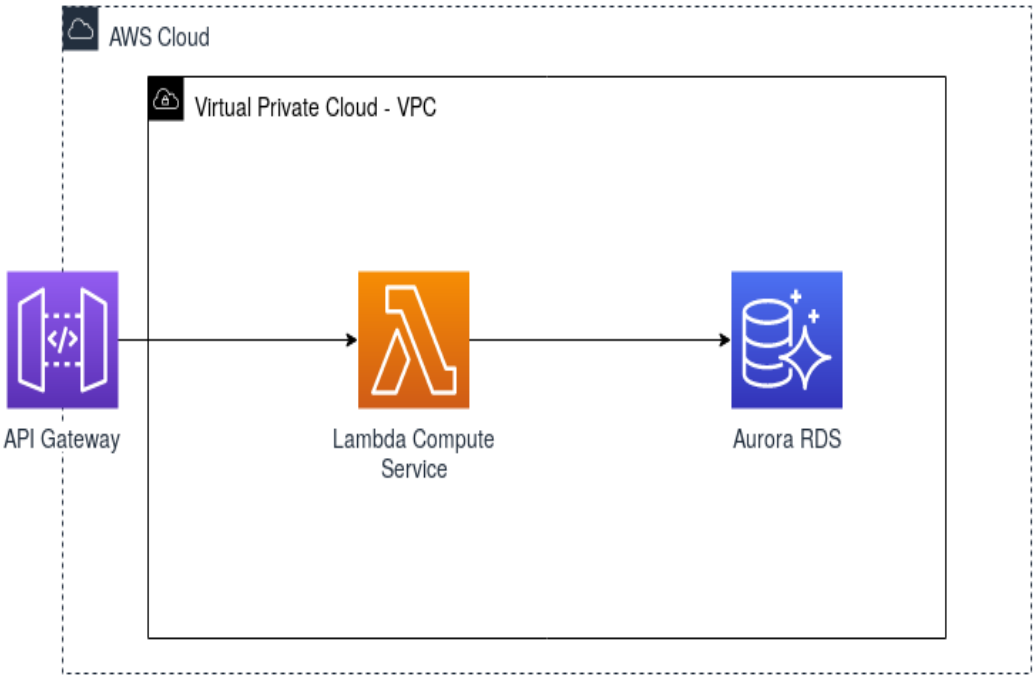
3.1. Αρχιτεκτονική Συστήματος

Η αρχιτεκτονική του παρόντος εγχειρήματος διαφέρει ελαφρώς από την τυπική διάταξη μιας υπηρεσίας / εφαρμογής ιστού και όπως θα δούμε αυτό οφείλεται στην υιοθέτηση της προσέγγισης "άνευ διακομιστών". Συνήθως μια cloud εφαρμογή περιλαμβάνει τρία επίπεδα (3 tiers). Ένα επίπεδο διακομιστών ιστού (web layer) οι οποίοι και είναι επιφορτισμένοι με την εξυπηρέτηση των αιτημάτων των χρηστών (client requests). Σε δεύτερο χρόνο οι τελευταίοι επικοινωνούν με ακόμα ένα επίπεδο διακομιστών (app layer) στους οποίους φιλοξενείται η εφαρμογή. Το τρίτο επίπεδο είναι αυτό των βάσεων δεδομένων (data layer) το οποίο προσπελάζεται από το επίπεδο της εφαρμογής κατά περίπτωση.

Στην περίπτωση της εφαρμογής άνευ διακομιστών η παραπάνω διάταξη των επιπέδων και στοιχείων δεν επιδέχεται ουσιαστικά καμία αλλαγή. Η πραγματική διαφοροποίηση βρίσκεται στην μορφή που παίρνουν τα συστατικά στοιχεία. Όπως εύκολα παρατηρεί κανείς τα βασικό συστατικό στοιχείο μιας εφαρμογής ιστού είναι οι διακομιστές. Επίσης εύκολα μπορεί να μαντέψει ότι στην αρχιτεκτονική άνευ διακομιστών έχουμε αντικατάσταση των συμβατικών διακομιστών με serverless υπηρεσίες που προσφέρονται από παρόχους Cloud. Έτσι στο πρώτο επίπεδο έχουμε πλέον μια υπηρεσία που αναλαμβάνει την "δημοσίευση" του API της εφαρμογής και την εξυπηρέτηση των αιτημάτων τα οποία και τελικά προωθούνται στο επόμενο επίπεδο. Αναφορικά με το δεύτερο επίπεδο η εφαρμογή είναι υλοποιημένη πλέον ως μια συνάρτηση η οποία δέχεται ως είσοδο το προωθούμενο αίτημα του πελάτη. Η συνάρτηση αυτή λοιπόν καλείται από την υπηρεσία που την φιλοξενεί για κάθε αίτημα

που δέχεται το πρώτο επίπεδο και με το πέρας της εκτέλεσής της επιστρέφει την έξοδο πάλι στο πρώτο επίπεδο για την τελική προώθηση στον πελάτη.

Όσον αφορά το τρίτο επίπεδο, αυτό των βάσεων δεδομένων, παραμένει ως έχει με την σημείωση ότι η διαχείριση και αποθήκευση των δεδομένων προσφέρεται και αυτή ως υπηρεσία η οποία είναι προσβάσιμη από το δεύτερο επίπεδο, την συνάρτηση-εφαρμογή. Μια γραφική αναπαράσταση της αρχιτεκτονικής του συστήματος δίνεται στο παρακάτω διάγραμμα:



Διάγραμμα 4. Αρχιτεκτονική πρωτοτύπου (1)

Όπως ήδη αναφέραμε στο κεφάλαιο 2 παρόλο που η αρχιτεκτονική άνευ διακομιστών είναι μια νέα τάση στην ανάπτυξη εφαρμογών και υπηρεσιών στο cloud υπάρχουν πολλοί πάροχοι (providers) που προσφέρουν εξελιγμένες και αξιόπιστες λύσεις όπως επίσης υπάρχουν και αρκετά πλαίσια ανάπτυξης (development frameworks) και εργαλεία ή/και βιβλιοθήκες για την απευθείας ανάπτυξη εφαρμογών με serverless τρόπο ή για την μετατροπή "συμβατικών" εφαρμογών σε serverless. Στο παρόν εγχείρημα χρησιμοποιούμε τις υπηρεσίες που προσφέρονται από τα Amazon Web Services αλλά παρόμοιες λύσεις προσφέρονται από όλους τους μεγάλους παρόχους. Η επιλογή της Amazon στηρίχτηκε στο γεγονός ότι είναι ο κορυφαίος πάροχος στην αγορά και οι Serverless λύσεις που προσφέρει χαίρουν μεγάλης δημοτικότητας και

επομένως υποστηρίζονται από μια μεγάλη κοινότητα η οποία συνεισφέρει με συμπληρωματικές λύσεις ή άλλες διευκολύνσεις. Στην συνέχεια θα εστιάσουμε την προσοχή μας σε κάθε ένα από τα στοιχεία της αρχιτεκτονικής δίνοντας μια περιγραφή της λειτουργίας και των χαρακτηριστικών τους.

AWS API Gateway

Η φιλοξενία του HTTP σημείου πρόσβασης, που αποτελεί και την είσοδο στην εφαρμογή, προσφέρεται από την υπηρεσία API Gateway. Η τελευταία χρησιμοποιείται για την δημιουργία, φιλοξενία και γενικότερη διαχείριση ή και ενίσχυση API (REST ή Websocket) εφαρμογών που βρίσκονται στο AWS cloud. Πρακτικά το API Gateway είναι επιφορτισμένο με την αποδοχή HTTP αιτημάτων από τους πελάτες (clients) και την προώθηση αυτών ως είσοδο στην υπηρεσία που φιλοξενείται η λογική της εφαρμογής μας.

AWS Lambda

Η Lambda είναι η Serverless υπολογιστική υπηρεσία της Amazon η οποία εκτελεί ένα προκαθορισμένο κομμάτι κώδικα σε απόκριση συμβάντων. Ο κώδικας που εκτελείται από την υπηρεσία ονομάζεται Lambda συνάρτηση. Μια τέτοια συνάρτηση μπορεί να δημιουργηθεί ανά πάσα στιγμή και να παραμετροποιηθεί κατά βούληση για μια σειρά υπολογιστικών μεγεθών και χαρακτηριστικών. Πιο συγκεκριμένα η υπηρεσία υποστηρίζει μια σειρά από γλώσσες προγραμματισμού κάνοντας χρήση διαφορετικών runtime systems (python, node.js κ.α.) όπως επίσης προσφέρει και δυνατότητες χρήσης custom runtimes ή μεγάλων εξωτερικών βιβλιοθηκών όπως γίνεται στο παρόν εγχείρημα για την χρήση των γεωχωρικών βιβλιοθηκών (για τις οποίες γίνεται λόγος παρακάτω).

Η λειτουργία της υπηρεσίας είναι οδηγούμενη από συμβάντα. Η Lambda διαθέτει την δυνατότητα να παρακολουθεί μια σειρά από πηγές (συμβάντων) – όπως για παράδειγμα ένα HTTP αίτημα από το API Gateway – και να καλεί μια συνάρτηση περνώντας ως είσοδο το ίδιο το συμβάν. Με το πέρας της εκτέλεσης του κώδικα η Lambda μπορεί επίσης να ρυθμιστεί να επιστρέψει την έξοδο της συνάρτησης πίσω στην πηγή – στο πρωτότυπό μας στο API Gateway το οποίο και την επαναπροωθεί στον πελάτη ολοκληρώνοντας έτσι τον κύκλο αιτήματος-απόκρισης (request-response cycle).

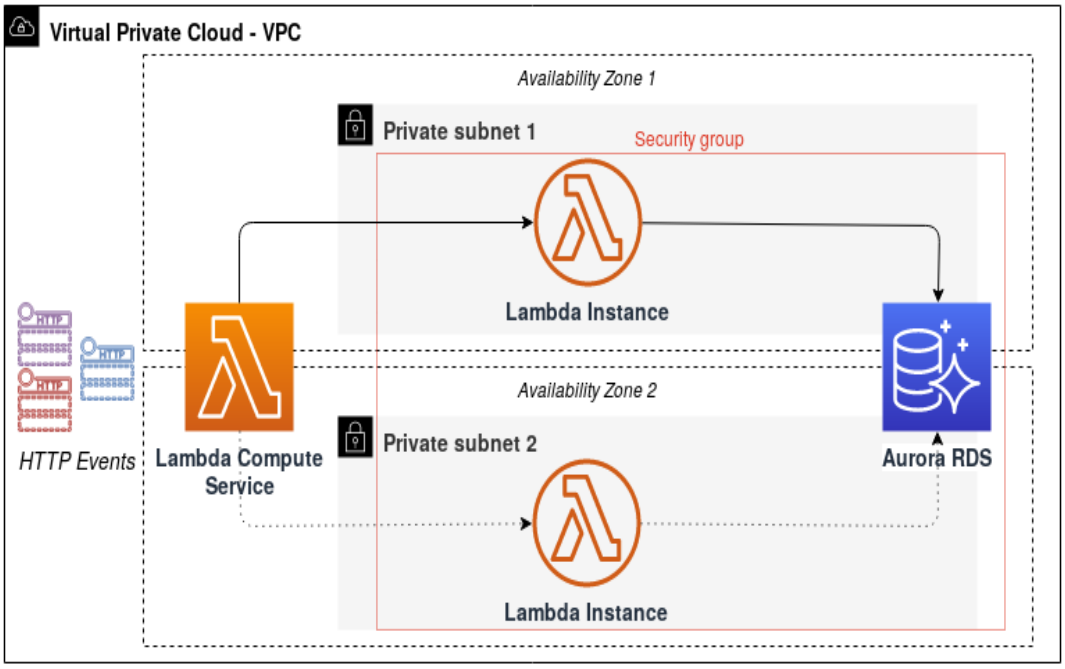
Αποτελώντας την υλοποίηση ενός FaaS, η Lambda αναλαμβάνει την διαχείριση των υποκείμενων υπολογιστικών πόρων προκειμένου η συνάρτηση να καλείται και ο κώδικας να εκτελείται σε κάθε συμβάν ανεξαρτήτως αριθμού και συχνότητας. Πρακτικά η υπηρεσία χρησιμοποιεί μεθόδους εικονικοποίησης για την δημιουργία αυτοτελών μικρο-εικονικών μηχανών (micro VMs) – τα στιγμιότυπα (Lambda instances) – οι οποίες στην ουσία είναι εφήμερα Linux συστήματα και περιλαμβάνουν τα απαραίτητα runtimes, επιλεγμένες (από τον προγραμματιστή-πελάτη) βιβλιοθήκες και φυσικά την συνάρτηση. Για κάθε συμβάν (στο πρωτότυπο μας για κάθε HTTP αίτημα) η πλατφόρμα της Lambda δημιουργεί ένα στιγμιότυπο το οποίο είναι ρυθμισμένο να καλεί την προκαθορισμένη συνάρτηση για να το επεξεργαστεί. Έτσι λοιπόν χάρη στην παραπάνω τεχνοτροπία η Lambda παρουσιάζει τα εξής χαρακτηριστικά:

- σχεδόν-μηδενικές ρυθμίσεις για την διάθεση και εκτέλεση κώδικα (near-zero configuration)
- υψηλή διαθεσιμότητα (high availability)
- αυτόματη και θεωρητικά απεριόριστη επεκτασιμότητα (infinite scalability)
- αυτόματη διαχείριση και ενημέρωση του υλικού και λογισμικού (μέχρι την συνάρτηση)
- χρέωση ανά millisecond

Επιστρέφοντας στην σκοπιά της αρχιτεκτονικής μπορούμε εύκολα να παρατηρήσουμε ότι η ροή των συμβάντων και πράξεων είναι στην ουσία η ίδια με αυτήν μιας κλασσικής (3-tier) αρχιτεκτονικής. Αυτό μπορεί εν μέρει να αποδοθεί στο γεγονός ότι μπορεί μεν η διάταξη και εκτέλεση της εφαρμογής να διαπεραιώνεται μέσω μιας “Serverless” συνάρτησης αλλά φυσικά η εκτέλεση της τελευταίας, όπως είδαμε αμέσως παραπάνω, πραγματοποιείται πάλι σε κάποιους διακομιστές (εικονικές μηχανές) των οποίων η λειτουργία υπόκεινται στους κλασσικούς φυσικούς και τεχνολογικούς περιορισμούς στους οποίους οφείλονται και τα μειονεκτήματα μιας FaaS υλοποίησης. Για παράδειγμα η δημιουργία στιγμιότυπων (για την επεξεργασία αιτημάτων) μπορεί να είναι μια διαδικασία αυτοματοποιημένη και επαρκώς βελτιστοποιημένη αλλά ακόμα απαιτεί αρκετά χιλιοστά του δευτερόλεπτου (έως και μερικά δευτερόλεπτα σε κάποιες περιπτώσεις) χαρακτηριστικό που καθιστά τέτοιες υπηρεσίες ακατάλληλες για

περιπτώσεις χρήσεις που περιλαμβάνουν διάδραση με τον τελικό χρήστη. Φυσικά λόγω της αυξανόμενης δημοτικότητας αλλά και των μεγάλων δυνατοτήτων που δίνει το Cloudta προαναφερθέντα μειονεκτήματα έχουν αμβλυνθεί (π.χ. το ίδιο στιγμιότυπο μπορεί να επεξεργαστεί περισσότερα του ενός αιτήματα χωρίς να απαιτείται εκ νέου αρχικοποίηση).

Ένα ακόμα παράδειγμα περιορισμού είναι το γεγονός ότι η επικοινωνία της εφαρμογής με τη βάση δεδομένων εξακολουθεί να υλοποιείται μέσω εικονικού δικτύου (VPC). Το τελευταίο μας δίνει μεν την δυνατότητα να ελέγχουμε πλήρως την κίνηση του δικτύου (network traffic) αλλά αυξάνει επίσης τον χρόνο αρχικοποίησης του στιγμιότυπου. Πριν προχωρήσουμε στο τρίτο και τελευταίο επίπεδο της αρχιτεκτονικής παραθέτουμε ένα πιο αναλυτικό διάγραμμα εστιασμένο στην λειτουργία της Lambda.



Διάγραμμα 5. Αρχιτεκτονική πρωτοτύπου (2)

AWS RDS Aurora (Serverless)

Η Relational Databse Service (RDS) είναι η υπηρεσία για τις (αυτο)διαχειριζόμενες (managed) βάσεις δεδομένων. Η τελευταία αναλαμβάνει την δημιουργία και φιλοξενία των πιο γνωστών RDBMS καθώς επίσης προσφέρει μια σειρά από εργασίες συντήρησης, προστασίας και επανόρθωσης δεδομένων.

Η Aurora είναι ένα DB engine αναπτυγμένο από την AWS το οποίο είναι συμβατό με τις πρόσφατες εκδόσεις της PostgreSQL και της MySQL. Διαφέρει από τις υπόλοιπα RDBMS ως προς την μέθοδο αποθήκευσης των δεδομένων. Ενώ οι συμβατικές μηχανές χρησιμοποιούν κλασσικές block storage συσκευές η Aurora απολαμβάνει ένα ειδικά κατασκευασμένο υψηλής επίδοσης υποσύστημα αποθήκευσης το οποίο δύναται να δημιουργεί πολλαπλά αντίγραφα των δεδομένων σε διαφορετικά μέρη εύκολα και γρήγορα. Το ιδιαίτερο όμως χαρακτηριστικό της είναι ότι προσφέρει την δυνατότητα να λειτουργεί σε Serverless mode. Αυτό την καθιστά μια από τις λίγες Serverless σχεσιακές βάσεις δεδομένων που έχει την δυνατότητα να "τρέχει" μόνο όταν το απαιτούν οι συνθήκες – πρακτικά όταν κάποια συνάρτηση (Lambda function) χρειάζεται να εκτελέσει κάποιο ερώτημα. Κατά το υπόλοιπο διάστημα η βάση μεταβαίνει σε κατάσταση παύσης (pause), στην οποία δεν καταναλώνει κανένα πόρο ελαχιστοποιώντας έτσι το λειτουργικό κόστος.

Η λειτουργία μιας βάσης αποτελεί συνήθως το μεγαλύτερο στοιχείο κόστους σε μία διάταξη. Ειδικά στις περιπτώσεις που είναι δύσκολο να προβλεφθεί η κίνηση που θα έχει η βάση ή σε περιπτώσεις που αναμένονται μεγάλες αυξομειώσεις η Serverless λειτουργία έχει σαφές πλεονέκτημα. Ένα ακόμα σενάριο που είναι ευνοϊκή είναι και η ανάπτυξη πρωτοτύπων στην οποία η βάση απαιτείται να είναι σε κανονική λειτουργία μόνο λίγες ώρες την ημέρα.

Αναφορικά με την μηχανή της βάσης δεδομένων επιλέγουμε την PostgreSQL[18]. Πρόκειται μια πολύ ισχυρή, ανοικτού κώδικα, αντικείμενο-σχεσιακή βάση δεδομένων με πάνω από 30 χρόνια ενεργούς ανάπτυξης από μια μεγάλη κοινότητα επιστημόνων, προγραμματιστών και υποστηρικτών και έχει αναπτύξει την φήμη μιας αξιόπιστης και υψηλών επιδόσεων λύσης στην αποθήκευση σχεσιακών δεδομένων. Η επιλογή μας βασίζεται τόσο στα προαναφερθέντα πλεονεκτήματα της PostgresSQL αλλά επίσης και σε μια επέκταση που η τελευταία διαθέτει, το PostGIS. Ενεργοποιώντας αυτή την επέκταση η PostgreSQL αναβαθμίζεται σε μια γεωχωρική βάση δεδομένων η οποία θα μας εξυπηρετήσει καλύτερα στην αποθήκευση και επεξεργασία των γεωχωρικών δεδομένων της εφαρμογής μας.

[18] https://www.postgresql.org/

3.2. Εργαλεία ανάπτυξης διαδικτυακής εφαρμογής

Έχοντας ολοκληρώσει την παρουσίαση και ανάλυση της αρχιτεκτονικής του συστήματος από την σκοπιά της υποδομής θα εστιάσουμε στα στοιχεία που απαρτίζουν την διαδικτυακή εφαρμογή μας σε επίπεδο λογισμικού.

3.2.1. Γλώσσα προγραμματισμού

Η επιλογή της κατάλληλης γλώσσας προγραμματισμού είναι κρίσιμη διότι προσδιορίζει την σχέση ανάμεσα στο πεδίο της εφαρμογής και στο πεδίο της έκφρασης. Δηλαδή, η δυνατότητα και η πλήρης αξιοποίηση στοιχείων του τομέα εφαρμογής μπορούν να γίνουν αντιληπτά ή σε πλήρη ανάπτυξη μόνο μέσα σε μια γλώσσα. Με δεδομένη την ύπαρξη πολλών διαφορετικών γλωσσών, αποτέλεσε σύνθετο ερευνητικό καθήκον η επιλογή μιας γλώσσας. Από τις επικρατούσες γλώσσες (Java, Javascript, Python, C#), επιλέχθηκε η Python[19] η οποία σε σχέση με τις άλλες πληροί μια σειρά από κριτήρια όπως:

- Στενή σχέση με λογισμικά GIS καθώς και πληθώρα πακέτων γεωχωρικής επεξεργασίας και ανάλυσης[20]
- Πρόκειται για μια γλώσσα φιλική προς το Cloud και ειδικότερα των Serverless διατάξεων κυρίως λόγω του ελαφρού διερμηνέα (interpeter) αλλά και του συνολικού runtime
- Πρόκειται για την ταχύτερα αναπτυσσόμενη γλώσσα προγραμματισμού
- Η χρήση της είναι αρκετά εύκολη όπως επίσης υπάρχει μεγάλη διαθεσιμότητα πόρων για την εκμάθηση της (https://www.jetbrains.com/lp/devecosystem-2019/)

Αξίζει να πραγματοποιηθεί μια σύντομη αναδρομή που δείχνει τον τρόπο και τις διαδικασίες μέσα από τις οποίες αναπτύχθηκε η εν λόγω γλώσσα. Η **Python** δημιουργήθηκε από τον (Guido van Rossum) στο ερευνητικό κέντρο Centrum Wiskunde & Informatica (CWI) το 1990. Ο Guido υπήρξε ο πρωταρχικός συγγραφέας της γλώσσας, αν και εμπεριέχει πολλές συνεισφορές και από άλλους. Το 2000 ο Guido

[19] https://www.python.org/
[20] https://brochure.getpython.info/

και η βασική ομάδα ανάπτυξης της γλώσσας δημιούργησαν την ομάδα BeOpen PythonLabs στην οποία και οφείλεται η κυκλοφορία της Python 2.0.

Ήδη από τις πρώτες κυκλοφορίες της γλώσσας και με την συνεργασία με προγραμματιστές εμπορικού λογισμικού διαφαινόταν η δυναμική να χρησιμοποιηθεί η γλώσσα σε λογισμικό που διατίθονταν υπό την άδεια GNU Public License (GPL). Έτσι, το CNRI και το Ίδρυμα Ελεύθερου Λογισμικού (FSF) συνεργάστηκαν προκειμένου να πραγματοποιηθούν οι κατάλληλες αλλαγές στην άδεια της γλώσσας οι οποίες και θα καταστήσουν τις επόμενες εκδόσεις της συμβατές με την άδεια GPL. Η Python 2.0.1 είναι το παράγωγο αυτής της εργασίας και της Python 2.0.

Το 2001 δημιουργείται το Ίδρυμα για το Λογισμικό της Python (Python Software Foundation) το οποίο και κατέχει όλα τα πνευματικά δικαιώματα της γλώσσας από την έκδοση 2.0 και έπειτα. Το ίδρυμα δημιουργήθηκε στα πρότυπα του ιδρύματος για το λογισμικό Apache (Apache Software Foundation), πρόκειται δηλαδή για ένα μη κερδοσκοπικό οργανισμό ο οποίος είναι ο μόνος φορέας επιφορτισμένος με την προώθηση, την προστασία και την εξέλιξη της γλώσσας Python καθώς επίσης και με την υποστήριξη και ανάπτυξη μιας διεθνούς και ποικίλης κοινότητας προγραμματιστών της.[21]

Από τεχνικής σκοπιάς πρόκειται για μια διερμηνευόμενη (interpreted), γενικού σκοπού (general-purpose) και υψηλού επιπέδου, γλώσσα προγραμματισμού. Ανήκει στις γλώσσες προστακτικού προγραμματισμού (*Imperative programming*) και υποστηρίζει τόσο το διαδικαστικό (*procedural programming*) όσο και το αντικειμενοστρεφές (*object-oriented programming*) προγραμματιστικό υπόδειγμα (*programming paradigm*). Είναι δυναμική γλώσσα προγραμματισμού (dynamically typed) και υποστηρίζει συλλογή απορριμμάτων (*garbage collection ή GC*).[22]

Κύριο σημείο στην φιλοσοφία της είναι ότι η αναγνωσιμότητα του κώδικά της και η ευκολία χρήσης της γλώσσας είναι εξίσου σημαντικά χαρακτηριστικά μιας γλώσσας. Ενδεικτικό είναι το γεγονός ότι σε αντίθεση με τις περισσότερες γνωστές γλώσσες η Python δεν κάνει χρήση άγκιστρων για τον διαχωρισμό τμημάτων (block) του κώδικα αλλά εσοχές (identation) με σκοπό την καλύτερη ταύτιση του κώδικα με την φυσική γλώσσα. Είναι επίσης γνωστή γιατί ενσωματώνει αρκετές βιβλιοθήκες στον βασικό

[21] https://en.wikipedia.org/wiki/History_of_Python
[22] https://en.wikipedia.org/wiki/Python_%28programming_language%29

κώδικα της γεγονός που οδήγησε αρκετούς επαγγελματίες της να χρησιμοποιούν την φράση “batteries included language runtime” .

Το βασικό μειονέκτημα της έγκειται στο γεγονός ότι είναι διερμηνευόμενη γλώσσα και επομένως πιο αργή από τις μεταγλωττιζόμενες (compiled) γλώσσες όπως η C και η C++ καθιστώντας την ακατάλληλη για συγκεκριμένα πεδία όπως για παράδειγμα την γραφή λειτουργικών συστημάτων.

Παρόλα αυτά οι διερμηνευτές της Python είναι διαθέσιμοι για εγκατάσταση σε πολλά λειτουργικά συστήματα, επιτρέποντας στην Python την εκτέλεση κώδικα σε ευρεία γκάμα συστημάτων. Χρησιμοποιώντας εργαλεία τρίτων, όπως το ή το Pyinstaller, ο κώδικας της Python μπορεί να πακεταριστεί σε αυτόνομα εκτελέσιμα προγράμματα για μερικά από τα πιο δημοφιλή λειτουργικά συστήματα, επιτρέποντας τη διανομή του βασισμένου σε Python λογισμικού για χρήση σε αυτά τα περιβάλλοντα χωρίς να απαιτείται εγκατάσταση του διερμηνευτή της Python.

3.2.2. Πλαίσιο ανάπτυξης – Django[23]

Έχοντας διαλέξει την γλώσσα συγγραφής της εφαρμογής το επόμενο βήμα είναι να επιλέξουμε ένα συμβατό πλαίσιο ανάπτυξης.

Τα πλαίσια ανάπτυξης εφαρμογών ιστού (Web Frameworks) είναι συλλογές πακέτων και βιβλιοθηκών που επιτρέπουν στους προγραμματιστές την συγγραφή εφαρμογών και υπηρεσιών ιστού χωρίς να διαθέτουν χρόνο για χαμηλού επιπέδου διεργασίες και λεπτομέρειες όπως παραδείγματος χάριν για δικτυακά πρωτόκολλα επικοινωνίας και διαχείριση νημάτων επεξεργασίας. Με απλά λόγια ένα πλαίσιο ανάπτυξης προσφέρει έτοιμες και εύχρηστες λύσεις στον προγραμματιστή για την αυτοματοποίηση διαδικασιών και διεργασιών που είναι συνηθισμένες αλλά αναγκαίες και επαναλαμβάνονται κατά την διάρκεια της αναπτυξιακής διαδικασίας σε κάθε έργο.

Ιστορικά η απαρχή τέτοιων πλαισίων τοποθετείται στις αρχές της δεκαετίας του 90’ με την καθιέρωση του προτύπου διεπαφής κοινής πύλης (Common Gateway Interface) για την διεκπεραίωση της επικοινωνίας μεταξύ ενός διακομιστή ιστού και μιας εξωτερικής εφαρμογής με απώτερο στόχο την διάθεση δυναμικών ιστοσελίδων. Η δεύτερη “γενιά” των πλαισίων εγκαινιάζεται με την εισαγωγή της PHP η οποία και λύνει αρκετά από τα

[23] https://www.djangoproject.com/

προβλήματα των παλαιών συστημάτων και με βασική καινοτομία την εύκολη κατανόηση και χρήση από αρχάριους προγραμματιστές. Από τότε τα web frameworks έχουν εξελιχθεί πάρα πολύ ενσωματώνοντας ακόμα περισσότερα εργαλεία υποστήριξης και λύνοντας μια σειρά από δομικά ζητήματα στα οποία οι προηγούμενης γενιάς τεχνολογίες δεν μπορούσαν να αντιμετωπίσουν. Τα σύγχρονα πλαίσια στοχεύουν στην αποφυγή της επανάληψης και στην προώθηση της επαναχρησιμοποίησης του κώδικα όπως επίσης παρέχουν προστασία από ευάλωτα σημεία σχετικά με την ασφάλεια της εφαρμογής. Προκειμένου να επιτευχθούν αυτοί οι στόχοι τα περισσότερα από αυτά ακολουθούν συγκεκριμένα αρχιτεκτονικά μοτίβα με πιο συνηθισμένο το μοντέλο-θέαση-ελεγκτής (Model-View-Controller – MVC). Το εν λόγω μοτίβο αποσκοπεί στον διαχωρισμό του μοντέλου των δεδομένων από την λογική της εφαρμογής καθώς και από την διεπαφή του χρήστη. Πιο αναλυτικά το μοτίβο διαχωρίζει τα εξής πεδία:

- Το μοντέλο (**M**) είναι μια αναπαράσταση των δεδομένων. Δεν είναι τα ίδια τα δεδομένα αλλά μια διεπαφή προς αυτά. Το μοντέλο επιτρέπει την μεταφορά των δεδομένων από και προς την βάση δεδομένων χωρίς αυτή η διαδικασία να εξαρτάται από τα συγκεκριμένα χαρακτηριστικά της βάσης. Στην ουσία το μοντέλο προσφέρει ένα επίπεδο αφαίρεσης από την βάση δεδομένων ώστε να καταστεί δυνατή η επαναχρησιμοποίηση του με διαφορετικές βάσεις δεδομένων.

- Η θέαση (**V**) είναι αυτό που βλέπουμε ως χρήστες. Πρόκειται για το επίπεδο της παρουσίασης του μοντέλου. Σε έναν υπολογιστή η θέαση είναι αυτό που βλέπουμε στον φυλλομετρητή μια εφαρμογής ιστούς ή στην γραφική διεπαφή χρήστη σε μια επιτραπέζια εφαρμογή (desktop app). Η θέαση επίσης παρέχει μια διεπαφή για την συλλογή της εισόδου του χρήστη.

- Ο ελεγκτής (**C**) ελέγχει την ροή της πληροφορίας ανάμεσα στο μοντέλο και την θέαση. Χρησιμοποιώντας προγραμματισμένη λογική γίνονται οι κατάλληλες επιλογές για το ποια πληροφορία ανακτάται από την βάση δεδομένων και διοχετεύεται στην θέαση. Επίσης είναι το στοιχείο που υλοποιεί την λογική της εφαρμογής συλλέγοντας την είσοδο του χρήστη και αξιολογώντας την διαφοροποιεί την θέαση ή αλλάζει τα δεδομένα ή και τα δύο.

Με αυτό τον τρόπο πραγματοποιείται η τμηματοποίηση του κώδικα με αποτέλεσμα την επαναχρησιμοποίηση τμημάτων, την ενίσχυση ευελιξίας της εφαρμογής αλλά και την ανεξάρτητη ανάπτυξη των τμημάτων (παράλληλη ανάπτυξη κ.α.).

Συνήθως τα πλαίσια ανάπτυξης εφαρμογών δομούνται γύρω από μια γλώσσα. Υπάρχουν αρκετά πλαίσια υλοποιημένα στην Python τα οποία ομαδοποιούνται σε δυο βασικές κατηγορίες με βάση τα χαρακτηριστικά που περιλαμβάνουν: τα πλήρη (full-stack) και τα μη πλήρη (non-full-stack).

Καθώς ο στόχος μας είναι να αναπτύξουμε μια fully-featured, standalone GIS εφαρμογή με γρήγορο και εύκολο τρόπο η επιλογή ενός πλήρους πλαισίου είναι πιο ταιριαστή. Τα πιο γνωστά Python Web Frameworks[24] αυτής της κατηγορίας είναι:

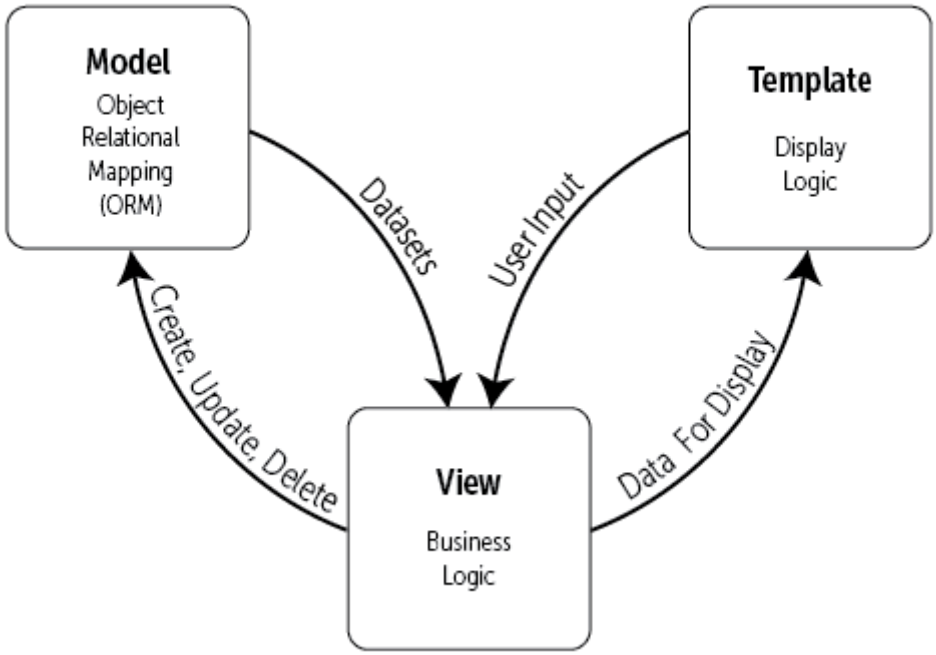
1. Django
2. TurboGears
3. web2py

Από αυτά το πιο δημοφιλές με αδιαμφισβήτητα την μεγαλύτερη κοινότητα είναι το Django. Το τελευταίο είναι από τις κυρίαρχες επιλογές για την δημιουργία εφαρμογών ιστού καθώς εκτός από την πολύ μεγάλη κοινότητα που το υποστηρίζει, προσφέρει αρκετά πλεονεκτήματα στους προγραμματιστές για ταχεία ανάπτυξη μοντέρνων εφαρμογών (και πρωτοτύπων) χωρίς περικοπές στην ασφάλεια ή τις προτεινόμενες προγραμματιστικές τεχνικές (best practices).

Παρόλο που το εν λόγω πλαίσιο ακολουθεί την δική του ορολογία, όπως η ονοματοθεσία "θέαση" στα τμήματα του κώδικα που επιστρέφουν HTTP αποκρίσεις, ο πυρήνας του ακολουθεί κατά βάση το μοτίβο MVC με κάποιες μικρές διαφοροποιήσεις. Πιο συγκεκριμένα το πλαίσιο δίνει βαρύτητα στο γεγονός ότι το στοιχείο "θέαση" αναφέρεται ρητά στο ποια δεδομένα (από το σύνολο των διαθέσιμων) επιλέγεται να επιστραφούν στον χρήστη κατόπιν κάποιου αιτήματος. Οι υποστηρικτές του Django φυσικά υποστηρίζουν τον διαχωρισμό της ίδιας της πληροφορίας (ή των επιλεγμένων δεδομένων) με τον τρόπο εμφάνισης τους δουλειά η οποία διεκπεραιώνεται από τα πρότυπα (templates). Έτσι από την οπτική του πλαισίου έχουμε πάλι τρία διακριτά στοιχεία με το ακρώνυμο του μοτίβου διαμορφώνεται σε μοντέλο-θέαση-πρότυπο (Model-View-Template MTV). Όσον αφορά τον ελεγκτή οι

[24] https://wiki.python.org/moin/WebFrameworks/

προγραμματιστές του Django ισχυρίζονται ότι πρόκειται για το σύνολο του πλαισίου και όλα εκείνα τα μέρη τα οποία επιτρέπουν την συνεργασία των προηγούμενων στοιχείων.



Εικόνα 3. Μοντέλο αρχιτεκτονικής MVC

Το πλαίσιο περιλαμβάνει ένα αντικείμενο-σχεσιακό χαρτογραφητή (object-relational mapper) ο οποίος μεσολαβεί μεταξύ των μοντέλων, τα οποία ορίζονται ως κλάσεις Python και μιας σχεσιακής βάσης δεδομένων, ένα σύστημα για επεξεργασία των HTTP αιτημάτων του οποίου τα κύρια μέρη είναι ένα σύστημα προτύπων ιστού και ένας αποστολέα URL βασισμένου σε κανονικές εκφράσεις (regural expressions).

Ένα χαρακτηριστικό του πλαισίου το οποίο μπορούμε να πούμε κληρονομεί και από την φιλοσοφία της ίδιας της γλώσσας πάνω στην οποία είναι βασισμένο είναι η πληθώρα βιβλιοθηκών και λειτουργιών που περιλαμβάνονται στον πυρήνα του λογισμικού και βρίσκονται άμεσα στην διάθεση του προγραμματιστή.

Έτσι εκτός από τα προαναφερθέντα στοιχεία ο πυρήνας ακόμα περιλαμβάνει:

- έναν ελαφρύ αλλά αυτόνομο διακομιστή κατάλληλο για την διαδικασία της ανάπτυξης αλλά και του testing μιας εφαρμογής

- ένα σύστημα σειριοποίησης (serialiazation) και επικύρωσης φορμών το οποίο επίσης φροντίζει την μετάφραση των τιμών από αυτές στην βάση και αντίστροφα
- ένα σύστημα προτύπων το οποίο χρησιμοποιεί το στοιχείο της κληρονομικότητας του αντικειμενοστραφούς προγραμματισμού
- ένα πλαίσιο το οποίο υποστηρίζει μια ποικιλία caching μεθόδων
- υποστήριξη για μια σειρά που διαμεσολαβούν μεταξύ διαφόρων σταδίων της διεκπεραίωσης των αιτημάτων και εκτελούν κώδικα κατά περίπτωση
- έναν σύστημα αποστολής το οποίο επιτρέπει στα διάφορα στοιχεία της εφαρμογής να επικοινωνούν μεταξύ τους με την χρήση προκαθορισμένων σημάτων
- ένα σύστημα διεθνοποίησης το οποίο περιλαμβάνει την μετάφραση των στοιχείων του Django σε μια ποικιλία γλωσσών
- ένα σύστημα σειριοποίησης το οποίο μπορεί να παράξει και να διαβάσει XML και JSON αναπαραστάσεις των μοντέλων
- ένα σύστημα επέκτασης των δυνατοτήτων της μηχανής των προτύπων
- μία διεπαφή για το ενσωματωμένο στην Python πλαίσιο για δοκιμές μονάδων (unit testing)

Ένα ακόμη ελκυστικό χαρακτηριστικό του Django framework είναι η επεκτασιμότητά του καθώς υποστηρίζει μια μεγάλη γκάμα επεκτάσεων (plugins) που ενσωματώνονται στην εκάστοτε εφαρμογή με ελάχιστες ρυθμίσεις. Σε ακριβώς αυτό το χαρακτηριστικό βασίζεται εν μέρει η επιλογή του πλαισίου για την τρέχουσα εφαρμογή καθώς υπάρχει η δυνατότητα χρήσης των γεωχωρικών δυνατοτήτων του πλαισίου απλώς ενεργοποιώντας τα αντίστοιχα modules για την επεξεργασία των γεωχωρικών δεδομένων όπως θα περιγράφουμε παρακάτω.

3.2.3. Επέκταση πλαισίου - GeoDjango

Το GeoDjango[25] είναι ένα module που συμπεριλαμβάνεται στο πλαίσιο στην κατηγορία των συνεισφερόμενων πακέτων / μονάδων και κατόπιν ενεργοποίησής του μετατρέπει το Django σε ένα πρώτης κλάσεως γεωγραφικό πλαίσιο ανάπτυξης εφαρμογών ιστού. Στόχος της μονάδας αυτής είναι η δημιουργία location-aware

[25] https://docs.djangoproject.com/en/3.0/ref/contrib/gis/

υπηρεσιών με όσο το δυνατών λιγότερη εργασία του προγραμματιστή. Ανάμεσα στα χαρακτηριστικά του περιλαμβάνονται:

- Πεδία των μοντέλων με υποστήριξη για γεωμετρικά (διανυσματικά) και Raster δεδομένων
- Επέκταση του υπάρχοντος ORM για την αναζήτηση και επεξεργασία των χωρικών δεδομένων
- Υψηλού επιπέδου Python διεπαφές για τις διάφορες τεχνικές και διαδικασίες γεωχωρικής ανάλυσης και υπολογισμών όσο και για την διαχείριση των δεδομένων σε διαφορετικές μορφές
- Δυνατότητα επεξεργασίας των γεωμετρικών πεδίων μέσα από την διεπαφή του διαχειριστή (built-in admin interface)

3.2.4. Γεωχωρικές βιβλιοθήκες

Για την εισαγωγή, διαχείριση και αποθήκευση των γεωχωρικών δεδομένων το GeoDjango στηρίζεται σε γεωχωρικές βιβλιοθήκες ανοικτού κώδικα. Πιο συγκεκριμένα το GeoDjango προσφέρει υψηλού επιπέδου Python διεπαφές για τις παρακάτω τρεις βιβλιοθήκες.

GEOS

Πρόκειται για την μετάφραση της JTS (επίσης γνωστή ως Java Topology Suite) βιβλιοθήκης στην γλώσσα C++ και στοχεύει στην υποστήριξη όλων των χωρικών συναρτήσεων και ενεργειών που περιγράφονται στο standard OpenGIS Simple Features for SQL του OGC.[26]

Η εν λόγω βιβλιοθήκη αποτελεί και το κύριο τρόπο αναπαράστασης και επεξεργασίας των γεωμετριών από το GeoDjango μέσω των υψηλού επιπέδου Python διεπαφών (lazy geometries) οι οποίες εκτελούν απευθείας κλήσεις στο C API της βιβλιοθήκης.

GDAL

Ονομασμένη από τα αρχικά των λέξεων: Geospatial Data Abstraction Library, η GDAL είναι μια βιβλιοθήκη-μεταφραστής τόσο για vector δεδομένα όσο και για raster.

[26] https://trac.osgeo.org/geos/

Παρέχοντας οδηγούς (drivers) για έναν μεγάλο αριθμό διαφορετικών τύπων γεωχωρικών δεδομένων χρησιμοποιείται από το πλαίσιο κυρίως για την αναγνώριση, ανάγνωση και εισαγωγή γεωχωρικών δεδομένων.[27]

PROJ

Πρόκειται για λογισμικό μετασχηματισμού γεωχωρικών συντεταγμένων από ένα γεωγραφικό σύστημα αναφοράς σε άλλο. Υποστηρίζει τόσο χαρτογραφικές προβολές όσο και σύνθετους γεωδαιτικούς μετασχηματισμούς.[28]

Συμπληρωματικά να αναφέρουμε ότι η ανάπτυξη της PROJ όπως και των άλλων δυο προαναφερθέντων βιβλιοθηκών είναι υπό την επίβλεψη του οργανισμού για την ανοικτότητα στα γεωχωρικά δεδομένα, Open Source Geospatial (OS Geo).

3.2.5. Serverless διάταξη - Zappa

Παρότι το Django χρησιμοποιείται για την ανάπτυξη μοντέρνων εφαρμογών ιστού δεν παρέχει την δυνατότητα διάταξης και εκτέλεσης με serverless τρόπο, χρησιμοποιώντας δηλαδή FaaS. Σε αυτό το πρόβλημα απαντάει μια νέα Python βιβλιοθήκη, η Zappa[29]. Η ανάπτυξη της τελευταίας ξεκίνησε στις αρχές του 2016 με πρωτοβουλία της ηλεκτρονικής πλατφόρμας διασύνδεσης open-source software freelancers, με στόχο την δυνατότητα μετατροπής WSGI-compatible εφαρμογών σε αρχιτεκτονική άνευ διακομιστών υποστηριζόμενες από τις υπηρεσίες Lambda και API Gateway της AWS.

Η βιβλιοθήκη έρχεται να συμπληρώσει και να ενισχύσει (enhance) την Django-developed (και γενικά όποια WSGI-compatible) εφαρμογή προσδίδοντάς τα χαρακτηριστικά – πλεονεκτήματα που εκφράζουν τις άνευ διακομιστή εφαρμογές.

Εκτός των άμεσων πλεονεκτημάτων που προσδίδονται στην εφαρμογή, η βιβλιοθήκη αυτή καθαυτή παρουσιάζει τα παρακάτω χαρακτηριστικά ενισχύοντας έμμεσα την ανάπτυξη της εφαρμογής:

- ευκολία στην χρήση (single command deployment)
- καμιά μετατροπή στον πηγαίο κώδικα της εφαρμογής (no lock-in)

[27] https://gdal.org/
[28] https://proj.org/
[29] https://github.com/Miserlou/Zappa

- ελεύθερο λογισμικό

3.3. Προγραμματιστική διεπαφή (API) - REST

Ο όρος REST (Representational state transfer) αναφέρεται σε ένα αρχιτεκτονικό στυλ για την σχεδίαση διαλειτουργικών υπολογιστικών συστημάτων στον παγκόσμιο ιστό. Οι υπηρεσίες που είναι συμβατές με το REST επιτρέπουν στα αιτούμενα συστήματα να έχουν πρόσβαση και να χειρίζονται τις αναπαραστάσεις των πόρων του διαδικτύου χρησιμοποιώντας ένα ομοιόμορφο και προκαθορισμένο σύνολο λειτουργιών που δεν συντηρούν κατάσταση.

Βασικό δομικό στοιχείο σε μια RESTful υπηρεσία είναι ο πόρος (resource) ο οποίος μπορεί να προσπελαστεί από κάποιον πελάτη (web client) πραγματοποιώντας ένα HTTP αίτημα. Ο εξυπηρετητής ανταποκρινόμενος στο αίτημα οφείλει να επιστρέψει πληροφορίες σχετικά με την κατάσταση του πόρου. Το περιεχόμενο της πληροφορίας όπως και η μορφή παρουσίασής της εξαρτάται από τις ειδικές παραμέτρους του αιτήματος. Μ άλλα λόγια το REST δίνει την δυνατότητα στο αιτούμενο σύστημα να αλληλεπιδράσει με διαφορετικό τρόπο με τον πόρο χρησιμοποιώντας διαφορετικό ρήμα (HTTP verb). Αναλυτικότερα το πρότυπο ορίζει το συντακτικό της επικοινωνίας ως “HTTP ρήμα” “πόρος”, με το νόημα των φράσεων αυτών να καταδεικνύει και το αναμενόμενο αποτέλεσμα των αιτημάτων. Για παράδειγμα το ρήμα GET ακολουθούμενο από το όνομα του πόρου επιστρέφει το σύνολο των στιγμιοτύπων (instances) για τα οποία διαθέτει πληροφορίες το αποκρινόμενο σύστημα και το ρήμα DELETE ακολουθούμενο από τον πόρο και ένα αναγνωριστικό διαγράφει το καταδεικνυόμενο στιγμιότυπο.

Το REST έχει γνωρίσει μεγάλη δημοτικότητα τα τελευταία χρόνια καθώς θεωρείται ο πιο μοντέρνος τρόπος δόμησης βιώσιμων, ευνόητων και αξιόπιστων API. Αυτά τα χαρακτηριστικά των RESTful API οφείλονται κυρίως στο γεγονός ότι το πρότυπο ορίζει αρκετά αυστηρά κριτήρια για την εφαρμογή του μεταξύ άλλων:

- Αρχιτεκτονική πελάτη – διακομιστή. Διαχωρισμός των δύο δρώντων με σκοπό την αυτόνομη και ανεξάρτητη ανάπτυξη και εξέλιξή τους.
- Χωρίς συντήρηση της κατάστασης. Τα αιτήματα που πραγματοποιούνται σε ένα RESTful API μεταφέρουν όλη την πληροφορία που χρειάζεται

προκειμένου να διεκπεραιωθούν. Με άλλα λόγια έαν εκτελέσουμε το ίδιο ερώτημα ν φορές θα πρέπει να παίρνουμε πάντα την ίδια απάντηση.

- Χρήση υπερμέσων (HATEOAS – Hypermedia As The Engine Of Application State). Η αρχή αυτή ορίζει ότι κάθε απάντηση του διακομιστή θα περιλαμβάνει και πληροφορίες για τους διαθέσιμους πόρους και τις ενέργειες με τις οποίες μπορεί να συνεχίσει το σύστημα-πελάτης.

Φυσικά ο σχεδιασμός και η υλοποίηση ενός RESTful API είναι αρκετά επίπονη και χρονοβόρα διαδικασία και όπως πολύ σωστά είχε αποφανθεί ο εμπνευστής του προτύπου: "οι άνθρωποι είναι αρκετά καλοί στους βραχυπρόθεσμους σχεδιασμούς και πολύ κακοί στους μακροπρόθεσμους". Μια πρόταση που πιστεύουμε ότι αποδίδει συνοπτικά και το αίτιο που πολλά συστήματα και υπηρεσίες ιστού ενώ σχεδιάστηκαν ακολουθώντας το πρότυπο κατέληξαν να παραβλέπουν πολλές από τις κατευθύνσεις του. Έτσι πολλές εφαρμογές και υπηρεσίες υιοθετούν μερική προσέγγιση του προτύπου ακολουθώντας μέρη των επιταγών με στόχο την ευκολότερη δημιουργία των API τους αλλά χωρίς επίσης να χάνουν όλα τα χαρακτηριστικά που προσέδιδε το REST.

Επιδιώκοντας την δημιουργία ενός βιώσιμου, διαλειτουργικού και επεκτάσιμου API για την εφαρμογή μας και χωρίς φυσικά να διαθέτουμε τους πόρους και την δυνατότητα για την δημιουργία ενός πλήρους RESTful API θα ακολουθήσουμε την προαναφερθείσα λογική του RESTlike API.

3.3.1. Επέκταση πλαισίου – Django REST Framework (DRF)[30]

Η επέκταση αυτή του Django αποτελεί ένα πολυ ισχυρό και ευέλικτο σετ εργαλείων για την δημιουργία προγραμματιστικών διεπαφών ιστού (Web APIs)

Μερικά από τα πλεονεκτήματα / χαρακτηριστικά της επέκτασης είναι:

- Ένα περιηγίσιμο μέσω ιστού API απευθείας με την εγκατάσταση της επέκτασης
- Πολιτικής αυθεντικοποίησης και υποστήριξη για OAuth1 και OAuth2
- Σειριοποίηση η οποία υποστηρίζει πηγές δεδομένων τόσο από ORM όσο και από μη-ORM

[30] https://www.django-rest-framework.org/

- Δυνατότητα χρήσης των ισχυρότατων κλάσεων των μοντέλων του Django ή παντελής παράκαμψη αυτών με χρήση κατά περίπτωση συναρτήσεων ή μίξη των δύο
- Ενεργή κοινότητα και καλά καταγεγραμμένες πληροφορίες χρήσης της επέκτασης

Φυσικά η επέκταση αυτή καθαυτή δεν παρέχει την αυτόματη δημιουργία RESTful APIs παρότι το όνομά της παραπέμπει σε κάτι τέτοιο, αποτελεί ωστόσο ένα πολύ βοηθητικό εργαλείο δημιουργίας διεπαφών ιστού για τις Django-developed εφαρμογές με μεγάλη ταχύτητα και ευκολία.

3.4. Εργαλεία ανάπτυξης γραφικής διεπαφής

Η ανάπτυξη της γραφικής διασύνδεσης έγινε χρησιμοποιώντας την δημοφιλή JavaScript βιβλιοθήκη ανάπτυξης διεπαφών χρήστη (UI building) React[31]. Η τελευταία δημιουργήθηκε και υποστηρίζεται από την γνωστή εταιρία social media facebook (υπό την εποπτεία της) και απολαμβάνει μια αρκετά μεγάλη κοινότητα προγραμματιστών και χρηστών.

Ενώ πλέον διατίθενται πολλές επιλογές εργαλείων λογισμικού για την ανάπτυξη διεπαφών οι πιο δημοφιλής και κατά συνέπεια με την μεγαλύτερη κοινότητα είναι οι: **Angular**, **Vue** και **React**. Ενώ ο στόχος της παρούσας εργασίας είναι αρκετά πιο ευρύς από την εξέταση της απόδοσης και καταλληλότητας κάποιου εργαλείου (UI framework), η επιλογή της React στηρίχτηκε κυρίως σε δύο χαρακτηριστικά της τελευταίας:

1) μικρό μέγεθος / lightweight
2) στενή σχέση με React Native.

Το πρώτο σημείο αποκτά ιδιαίτερη βαρύτητα στην περίπτωση μας δεδομένης της serverless αρχιτεκτονικής επιλογής – έκθεση & εκτέλεση κώδικα σε περιβάλλον με αρκετούς περιορισμούς πόρων. Το δεύτερο πλεονέκτημα αφορά την ευκολία "μεταφοράς" / επέκτασης της εφαρμογής σε κινητή έκδοση (mobile app) - ιδιότητα η

[31] https://reactjs.org/

οποία λόγω και του πεδίου της εφαρμογής (αγροτικές καλλιέργειες) είναι καθοριστικής σημασίας για την χρηστικότητά της.

Επιπλέον ένα βασικά χαρακτηριστικά της βιβλιοθήκης είναι ο κατακερματισμός της διεπαφής σε συστατικά στοιχεία (components) τα οποία δρουν αυτόνομα και τοιουτοτρόπως επιτυγχάνεται αυξημένη ευελιξία και επίδοση καθώς και καλύτερη οργάνωση / συντήρηση του κώδικα.

3.4.1. Βιβλιοθήκη διαδικτυακών χαρτών - Leaflet

Μια από τις πιο δημοφιλής JavaScript βιβλιοθήκες ανοικτού-κώδικα για την δημιουργία χαρτών ιστού είναι η Leaflet[32]. Όπως η περιγραφή της ιστοσελίδας της αναφέρει: ο πρωταρχικός της στόχος είναι η δημιουργία mobile-friendly διαδραστικών χαρτών. Η βιβλιοθήκη δημιουργήθηκε από τον Vladimir Agafokin[33] αλλά η ανάπτυξή της πλέον βασίζεται σε μια πολυάριθμη και ενεργή κοινότητα προγραμματιστών. Η βιβλιοθήκη αναπτύσσεται στους άξονες της απλότητας, δυνατών επιδόσεων και χρηστικότητας χαρακτηριστικά που τιμά καθώς αφενός το μέγεθος του κώδικα της δεν ξεπερνά τα 38KB και αφετέρου η δημιουργία και εκτέλεση απλών και πιο σύνθετων γεωχωρικών στοιχείων και υπολογισμών γίνεται με ευκολία καθώς προσφέρει φιλικές και καλά επεξηγούμενες δομές. Άλλες δημοφιλείς βιβλιοθήκες είναι: η OpenLayers, η MapBox GL JS και η Datamaps.

Λόγω της ευρείας αποδοχής της βιβλιοθήκης και των πολυάριθμων χρηστών της η τελευταία χαίρει μεγάλο αριθμό αξιόπιστων επεκτάσεων. Ειδικότερα στο πρωτότυπό μας κάνουμε χρήση του *react-leaflet*[34] και του *Leaflet-Geoman*[35]. Το πρώτο επιτρέπει αφαιρέσεις της βιβλιοθήκης ως συστατικά (components) της react ενώ το δεύτερο επιτρέπει την εύκολη και πλούσια σε στοιχεία σχεδίαση και επεξεργασία των γεωμετρικών επιπέδων του χάρτη.

[32] https://leafletjs.com/
[33] https://agafonkin.com/
[34] https://react-leaflet.js.org/
[35] https://github.com/geoman-io/leaflet-geoman

4. Υλοποίηση

Έχοντας ολοκληρώσει την περιγραφή του προτεινόμενου συστήματός και έχοντας επιλέξει τα εργαλεία που θα χρησιμοποιήσουμε θα προχωρήσουμε στην υλοποίηση της εφαρμογής.

Όπως αναφέραμε και στην εισαγωγή η κύρια λειτουργία της εφαρμογής είναι η συλλογή και αποθήκευση περιγραφικών δεδομένων αγροτεμαχίων και συνδέοντας τα με γεωχωρική πληροφορία με απώτερο σκοπό την εκτέλεση υπολογισμών και αναλύσεων και την μετέπειτα τεκμηριωμένη εξαγωγή συμπερασμάτων για μια σειρά θεμάτων σχετικά με τις αγροτικές καλλιέργειες ανά την επικράτεια.

Δεδομένου του παραπάνω στόχου θα επιχειρήσουμε παρακάτω να σκιαγραφήσουμε τις βασικές απαιτήσεις που θα πρέπει να καλύπτει το πρωτότυπο

- Δυνατότητα δημιουργίας (και προβολής) ψηφιακής απεικόνισης του αγροτεμαχίου με προσδιορισμό της τοποθεσίας του
- Ο προσδιορισμός της τοποθεσίας του αγροτεμαχίου να μπορεί να συμβεί με διαφορετικούς τρόπους:
    - Καταχώριση γεωγραφικών συντεταγμένων του τεμαχίου (π.χ. από τοπογραφικό ή άλλη πληροφορία) χρησιμοποιώντας μια REST-like διεπαφή. Η τελευταία θα πρέπει να υποστηρίζει το GeoJSON format για την αποθήκευση και ανάκτηση της γεωχωρικής πληροφορία.
    - Χρήση (ανέβασμα) κατάλληλου αρχείου που να ενσωματώνει επαρκή πληροφορία (shapefile) για την απεικόνιση του τεμαχίου σε χάρτη
    - Επιλογή περιοχής στον χάρτη της εφαρμογής (GUI only)
- Δημιουργία (και προβολή) ψηφιακής απεικόνισης καλλιεργειών (των καταχωρημένων τεμαχίων) με διανυσματικό τρόπο
- Δυνατότητα καταχώρισης περιγραφικών πληροφοριών για μια καλλιέργεια με ταυτόχρονη δυνατότητα την συμπερίληψη γεωχωρικής πληροφορίας όποτε είναι αυτό δυνατό. Αναλυτικότερα θα πρέπει να παρέχεται η δυνατότητα να καταχωρούνται πάσης φύσεως συμβάντα που αφορούν την καλλιέργεια και για τα οποία θα δίνεται η δυνατότητα καταχώρισης της πληροφορίας με την (ταυτόχρονη ή όχι) χρήση των παραπάνω μεθόδων.

- Ανάκτηση βασικών γεωχωρικών πληροφοριών των καλλιεργειών και των αγροτεμαχίων (εμβαδόν, περίμετρο κλπ)

Θεωρούμε ότι η εφαρμογή θα πρέπει να ανταποκρίνεται επίσης και σε μη λειτουργικές απαιτήσεις στους τομείς της απόδοσης, ασφάλειας, διαθεσιμότητας αλλά και προσβασιμότητας και πρακτικότητας με την μέση επίδοση των εφαρμογών στον κλάδο και φυσικά πάντα τηρούμενων των οδηγιών και κανονισμών περί περιφρούρησης ιδιωτικών και ευαίσθητων δεδομένων. Παρότι δεν προβλέπονται ειδικές απαιτήσεις στόχος της υλοποίησης είναι η διατήρηση του κόστους (χρηματικό και εργατοωρών) ανάπτυξης, διάταξης και δοκιμαστικής λειτουργίας του πρωτοτύπου να παραμένει στο ελάχιστο..

4.1. Μοντέλα

Τα μοντέλα αποτελούν μια αφηρημένη δομή η οποία οργανώνει τα δεδομένα με τέτοιο τρόπο ώστε να αναπαριστούν ιδιότητες των οντοτήτων του πραγματικού κόσμου με απώτερο σκοπό την προβολή στοιχείων του τελευταίου στον ψηφιακό. Αφού λοιπόν δεν είναι παρά αναπαραστάσεις πραγματικών οντοτήτων, τα μοντέλα περιλαμβάνουν όλα τα πεδία και τις συμπεριφορές των πρώτων. Στον αντικειμενοστραφή προγραμματισμό η δημιουργία των μοντέλων γίνεται με την διατύπωση μιας κλάσης, τα στιγμιότυπα της οποίας ενθυλακώνουν όλη την πληροφορία σχετικά με την οντότητα που αναπαριστούν καθώς επίσης και την διαχείριση αυτής (π.χ. τρόπος και τόπος αποθήκευσης ή ανάκτησης πληροφορίας).

Κατά την αποθήκευση των πληροφοριών ένα μοντέλο (ή μια κλάση) αντιστοιχίζονται με έναν πίνακα σε μια σχεσιακή βάση δεδομένων, οι εγγραφές τις οποίας αποτελούν τα διάφορα στιγμιότυπα του μοντέλου. Τυπικά οι πληροφορίες και οι διαδικασίες εισόδου και εξόδους αυτής πρέπει να περιγράφουν στην κλάση με όλες τις σχετικές λεπτομέρειες αλλά καθώς αυτό εμπεριέχει μεγάλο βαθμό επανάληψης κώδικα τα σύγχρονα πλαίσια ανάπτυξης αναλαμβάνουν να απλοποιούν την περιγραφή ενός μοντέλου / κλάσης στα πολύ ειδικά χαρακτηριστικά μιας εφαρμογής (application specific).

Για τον προσδιορισμό ενός μοντέλου στο GeoDjango μπορούμε να κάνουμε χρήση έτοιμων κατηγοριών πεδίων οι οποίες ενσωματώνουν όλους τους απαραίτητους

ελέγχους εισόδου της πληροφορίας. Το ακόμα μεγαλύτερο πλεονέκτημα χρήσης του πλαισίου στην δημιουργία των μοντέλων είναι ο ενσωματωμένος μηχανισμός ORM (Object-Relational Mapping). Ο τελευταίος υποστηρίζει όλες τις διαδικασίες δημιουργίας και τροποποίησης των πινάκων της βάσης δεδομένων που προορίζονται για την αποθήκευση των δεδομένων των μοντέλων (scaffolding) όπως επίσης και όλες τις διαδικασίες για την είσοδο / έξοδο της πληροφορίας από τα μοντέλα προς τη βάση και αντίστροφα. Έτσι για την πλήρη περιγραφή των μοντέλων μας αρκούν μερικές γραμμές κώδικα οι οποίες αναφέρονται στα ιδιαίτερα χαρακτηριστικά τους.

Ακολουθεί η περιγραφή των μοντέλων της εφαρμογής και η υλοποίηση τους.

Γεωτεμάχιο - Field

Η αναπαράσταση τμήματος ενός γεωτεμαχίου το οποίο βρίσκεται υπό την κατοχή ή απλώς χρήση από τον χρήστη-αγρότη. Το παρόν θα αποτελέσει το πρώτο διανυσματικό επίπεδο πάνω στο οποίο θα εκτελούνται οι περισσότεροι γεωχωρικοί υπολογισμοί. Επίσης θα αποτελέσει και το βασικό πεδίο αναφοράς κατά την εισροή πληροφορίας (περιγραφικών δεδομένων ή επιπλέον διανυσματικών επιπέδων) από τον χρήστη ή από άλλα συστήματα.

Το μοντέλο του γεωτεμαχίου στην τρέχουσα υλοποίηση έχει δύο πεδία όπως φαίνεται και από την εικόνα 4. Το πρώτο είναι ένα όνομα για την ευκολότερη ταυτοποίηση του ή αναφορά σε αυτό από το χρήστη. Το δεύτερο είναι η τοποθεσία στην οποία θα αποθηκεύεται η γεωχωρική πληροφορία για μια έκταση. Το πεδίο αυτό είναι τύπου *MULTIPOLYGON* και υποστηρίζει διανύσματα πολύγωνου ή πολλαπλών πολυγώνων σε γεωγραφικό σύστημα αναφοράς WGS 84 (SRID: 4326).

Στην κλάση του γεωτεμαχίου επίσης υλοποιούνται και οι μέθοδοι για τον υπολογισμό βασικών γεωχωρικών μεγεθών (επιφάνεια, περίμετρος, κεντροϊδές). Οι υπολογισμοί αυτοί όπως και άλλες τοπολογικού ενδιαφέροντος πληροφορίες είναι δυνατοί και μπορούν να εκτελεστούν σε κάθε γεωχωρικό πεδίο με φυσικό τρόπο (natively) χάρις στις διεπαφές που προσφέρει το GeoDjango για την βιβλιοθήκη *GEOS*.

```
6
7   class Field(models.Model):
8       name = models.CharField(max_length=200)
9       location = gis_models.MultiPolygonField(srid=4326, null=False, blank=False)
10
11      def calc_area(self):
12          polygon = self.location.transform(2100, clone=True)
13          area = round(polygon.area, 2)
14
15          return area
16
17      def calc_perimeter(self):
18          polygon = self.location.transform(2100, clone=True)
19          length = round(polygon.length, 6)
20
21          return length
22
23      def calc_centroid(self):
24          centroid = self.location.centroid
25
26          return centroid
27
28      def __str__(self):
29          return '%s' %(self.name)
30
31      def __unicode__(self):
32          return '%s' %(self.name)
33
```

Εικόνα 4. Μοντέλο γεωτεμαχίου

Καλλιέργεια - Cultivation

Η αναπαράσταση της καλλιέργειας ως ένα πολύγονο το οποίο θα πρέπει να βρίσκεται εντός ενός υπαρκτού / δηλωμένου γεωτεμαχίου. Το μοντέλο εμπεριέχει την πληροφορία σχετικά με μια καλλιέργεια Όπως είναι φυσικό η εν λόγω κλάση παρουσιάζει πολλές ομοιότητες με την προηγούμενη. Όπως φαίνεται και στην εικόνα 5 η καλλιέργεια έχει τα ίδια πεδία με το γεωτεμάχιο και επιπλέον έχει δυο ακόμα πεδία για την κατηγοριοποίηση της και την σύνδεση της με ένα γεωτεμάχιο. Σε ότι αφορά τις μεθόδους υπάρχουν και εδώ οι προαναφερθείσες μέθοδοι υπολογισμού και επιστροφής βασικών γεωχωρικών μεγεθών και ακόμα η υποσκέλιση της μεθόδου για την αποθήκευση του αντικειμένου στην βάση προκειμένου να υλοποιήσουμε τον έλεγχο της συμπερίληψης της καλλιέργειας σε ένα δηλωμένο γεωτεμάχιο. Ξανά η εκτέλεση αυτού του ελέγχου, επί της ουσίας ενός γεωχωρικού υπολογισμού υλοποιείται απλώς με την επίκληση της μεθόδου των πολυγώνων “within” όπως φαίνεται στην γραμμή 62, μέθοδος η οποία στην ουσία στηρίζεται στην βιβλιοθήκη GEOS που όπως ήδη αναφέραμε χρησιμοποιεί το GeoDjango για την αναπαράσταση των γεωμετριών.

```python
cultivation_types = [('apples', 'Apples'), ('flowers', 'Flowers'), ('olives', 'Olives')]


class Cultivation(models.Model):
    name = models.CharField(max_length=200)
    location = gis_models.MultiPolygonField(srid=4326, null=False, blank=False)
    category = models.CharField(max_length=20, choices=cultivation_types, default='site')
    field = models.ForeignKey(Field, on_delete=models.CASCADE)

    def calc_area(self):
        polygon = self.location.transform(2100, clone=True)
        area = round(polygon.area, 2)

        return area

    def calc_perimeter(self):
        polygon = self.location.transform(2100, clone=True)
        length = round(polygon.length, 6)

        return length

    def calc_centroid(self):
        centroid = self.location.centroid

        return centroid

    def save(self, *args, **kwargs):
        is_inside_field = self.location.within(Field.objects.get(pk=self.field.id).location)

        if(is_inside_field):
            super(Cultivation, self).save(*args, **kwargs)
        else:
            raise ValidationError('Input geometry is not included in any of the Fields registered')

    def __str__(self):
        return '%s' %(self.name)

    def __unicode__(self):
        return '%s' %(self.name)

```

Εικόνα 5. Μοντέλο καλλιέργειας

Σημεία Ενδιαφέροντος – Points of Interest

Το τελευταίο μοντέλο ενός εφαρμογής είναι τα σημεία ενδιαφέροντος. Με αυτό το μοντέλο επιδιώκουμε την αναπαράσταση:

1) Συμβάντων σχετικών με την καλλιέργεια (π.χ. μέτρηση περιβαλλοντικών μεταβλητών ή μεγεθών σχετικών με την καλλιέργεια, αναφορά / καταγραφή κάποιας ασυνήθιστης κατάστασης). Η διάκριση αυτών γίνεται με τον τύπο “Event” και με τον τύπο “Measurement”

2) Σημεία ειδικού ενδιαφέροντος (π.χ. σημείο υδροληψίας) – τύπος “Site”

Τα παραπάνω μπορούν να αποτυπωθούν σημειακά και η προϋπόθεση δημιουργίας ενός σημείου ενδιαφέροντος (Point of Interest) είναι η συμπερίληψη του εντός ενός γεωτεμαχίου.

```
point_of_interest_types = [('event', 'Event'), ('measurement', 'Measurement'), ('site', 'Site')]


class PointOfInterest(models.Model):
    name = models.CharField(max_length=200)
    location = gis_models.PointField(srid=4326, null=False, blank=False)
    field = models.ForeignKey(Field, on_delete=models.CASCADE)

    description = models.TextField(null=True)
    category = models.CharField(max_length=20, choices=point_of_interest_types, default='site')
    humidity = models.IntegerField(null=True)
    luminocity = models.FloatField(null=True)
    temperature = models.FloatField(null=True)

    def save(self, *args, **kwargs):
        is_inside_field = self.location.within(Field.objects.get(pk=self.field.id).location)

        if(is_inside_field):
            super(PointOfInterest, self).save(*args, **kwargs)
        else:
            raise ValidationError('Input geometry is not included in any of the Fields registered')

    def __str__(self):
        return '%s' %(self.name)

    def __unicode__(self):
        return '%s' %(self.name)

```

Εικόνα 6. Μοντέλο σημείου ενδιαφέροντος

Σε αντίθεση με τα προηγούμενα δύο μοντέλα χωρική πληροφορία του σημείου ενδιαφέροντος είναι ένα προφανώς ένα σημείο (*POINT*) και επίσης περιλαμβάνει πεδία για την αποθήκευση επιπλέον περιγραφικών δεδομένων σχετικές με το συμβάν ή την μέτρηση (υγρασία, θερμοκρασία, φωτεινότητα). Όπως και στην καλλιέργεια έτσι και εδώ έχουμε την σύνδεση με κάποιο κτήμα όπως και την υποσκέλιση του κατασκευαστή για την υλοποίηση του ελέγχου περιβολής του σημείου από τα όρια κάποιου κτήματος.

4.2 Προγραμματιστική διεπαφή εφαρμογής ιστού

4.2.1. REST-Like API

Ο δίαυλος επικοινωνίας μιας εφαρμογής ιστού με τον «έξω κόσμο είναι μια προγραμματιστική διεπαφή (API) η οποία χρησιμοποιεί και υποστηρίζει τα πρωτόκολλα του ιστού για την μεταφορά των πληροφοριών από και προς την εφαρμογή. Όπως ήδη περιγράψαμε στα εργαλεία ανάπτυξης στην τρέχουσα υλοποίηση επιλέξαμε το αρχιτεκτονικό στυλ REST για την διαμόρφωση της διεπαφής. Με απλά λόγια το προηγούμενο σημαίνει ότι η εφαρμογή εκθέτει σημεία πρόσβασης (endpoints) μέσω των οποίων μπορεί κάποιος εξωτερικός χρήστης ή ένα σύστημα να αλληλεπίδρασει με τους πόρους της εφαρμογής. Οι τελευταίοι είναι συνήθως τα μοντέλα της εφαρμογής.

Οι ενέργειες που εκτελούνται πάνω σε κάποιον πόρο είναι τυπικά τέσσερις: 1) ανάκτηση, 2) δημιουργία, 3) τροποποίηση, 4) διαγραφή ευρύτερα γνωστές ως CRUD (Create, Retrieve, Update, Delete). Ένα πελάτης που επιθυμεί να αλληλεπιδράσει με κάποιον πόρο λοιπόν χρειάζεται να στείλει ένα HTTP αίτημα προσδιορίζοντας τον πόρο συνήθως μέσω του URI (μοναδική διαδρομή για τον πόρο), την ενέργεια μέσω του HTTP ρήματος που θα χρησιμοποιήσει και ενδεχομένως ένα σώμα πληροφοριών σχετικές με τον πόρο και την ενέργεια (body / payload). Για τα REST-like APIs το σώμα τυπικά είναι σε μορφή JSON – μορφή η οποία μπορεί να μεταφερθεί και να επεξεργαστεί από οποιοδήποτε σύστημα ικανό να χρησιμοποιήσει τον ιστό.

Αφότου ένα αίτημα φτάσει στην διεπαφή υποβάλλεται σε ένα στάδιο αποσειριοποίησης (deserialization) κατά το οποίο τα δεδομένα του αιτήματος και κυρίως το σώμα μετατρέπεται από JSON σε δομές δεδομένων «κατανοητών» για την εφαρμογή. Σε δεύτερο στάδιο οι δομές δεδομένων με τυχόν άλλες πληροφορίες περνούν ως είσοδο στο τμήμα της βασικής λογικής η οποία τις επεξεργάζεται και εκτελεί την ενέργεια που ζητήθηκε.

Όπως και στα τα μοντέλα έτσι και η προαναφερθείσα λειτουργικότητα της διεπαφής περιλαμβάνει αρκετά τετριμμένες διαδικασίες οι οποίες οδηγούν σε επανάληψη του κώδικα εα δημιουργηθούν εξ’ αρχής για κάθε πόρο. Ξανά στο πρόβλημα αυτό τα πλαίσια ανάπτυξης δίνουν την λύση παρέχοντας στον προγραμματιστή έτοιμες κλάσεις και δομές για να ελαχιστοποιηθεί ο κώδικας που απαιτείται για την υλοποίηση της διεπαφής.

Η υλοποίηση της διεπαφής πραγματοποιείται χρησιμοποιώντας την επέκταση του *Django*: *Django Rest Framework (DRF)* η οποία επιτρέπει τον προσδιορισμό σειριοποιητών (serializers) για την μετατροπή των δεδομένων ενός εξωτερικού αιτήματος σε δομές και τύπους συμβατούς με τα μοντέλα της εφαρμογής. Στο πρωτότυπο μας έχουμε ορίσει τρεις βασικούς σειριοποιητές, έναν για κάθε μοντέλο συν τρεις βοηθητικούς. Ενδεικτικά παραθέτουμε τον κώδικα ενός από αυτούς στην παρακάτω εικόνα καθώς οι υπόλοιποι είναι υλοποιημένοι με παρόμοιο τρόπο.

```python
class FieldSerializer(GeoFeatureModelSerializer):
    surface = serializers.SerializerMethodField()
    circumference = serializers.SerializerMethodField()

    class Meta:
        model = Field
        geo_field = 'location'
        fields = ['id', 'name', 'surface', 'circumference']

    def get_surface(self, instance):

        try:
            area = instance.calc_area()
        except Exception as e:
            print('Exception caught while calculating surface!', str(e))

            return 'UNABLE_TO_CALCULATE'

        return area

    def get_circumference(self, instance):

        try:
            perimeter = instance.calc_perimeter()
        except Exception as e:
            print('Exception caught while calculating circumference!', str(e))

            return 'UNABLE_TO_CALCULATE'

        return perimeter

```

Εικόνα 7. Σειριοποιητής γεωτεμαχίου

Όπως φαίνεται στην εικόνα 7 για τον ορισμό του προσδιορίζουμε το μοντέλο δεδομένων και τα πεδία που επιδιώκουμε να μεταφερθούν. Μια επιπλέον δυνατότητα των σειριοποιητών του DRF είναι η δημιουργία υπολογιζόμενων πεδίων τα οποία δεν είναι αποθηκευμένα στην βάση αλλά προκύπτουν καθώς επεξεργαζόμαστε άλλα. Παράδειγμα χρήσης αυτής της δυνατότητας είναι η εμφάνιση της επιφάνειας και της περιμέτρου ως πεδία των μοντέλων. Όσον αφορά την παρουσίαση και μετατροπή της μορφής της χωρικής πληροφορίας κάνουμε χρήση της βιβλιοθήκης *django-rest-framework-gis*[36] η οποία στην ουσία επεκτείνει το DRF προσδίδοντας στους σειριοποιητές την δυνατότητα αναγνώρισης, ανάλυσης και μετατροπής WKT μορφής χωρική πληροφορία σε GeoJSON και αντίστροφα. Η δυνατότητα αυτή ενεργοποιείται για το πεδίο που θα ορίσουμε ως *geo_field (εικόνα 7 γραμμή 14)*.

Η δεύτερη αλλά εξίσου σημαντική δυνατότητα που μας παρέχεται από το DRF είναι οι έτοιμες κλάσεις για εκτέλεση λογικής CRUD επάνω στα μοντέλα. Με αυτόν τον τρόπο

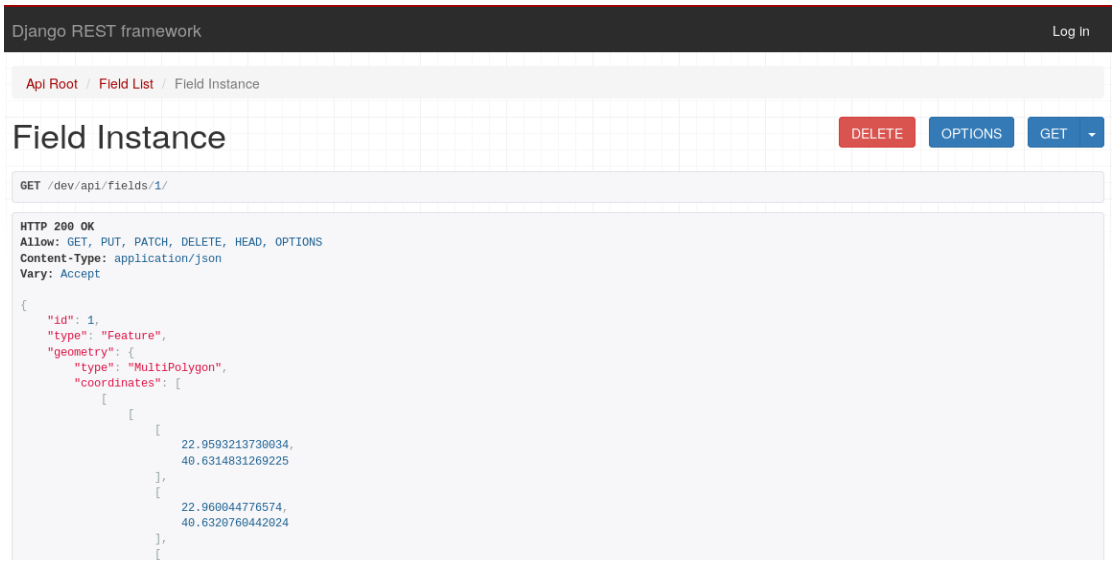
[36] https://github.com/openwisp/django-rest-framework-gis

όλος ο κώδικας που απαιτείται για την υλοποίηση CRUD λειτουργικότητας τους API είναι μόνο ο προσδιορισμός ενός σειριοποιητή και ενός συνόλου αντικειμένων όπως φαίνεται και στην εικόνα 8.

```
32  class FieldViewSet(viewsets.ModelViewSet):
33      queryset = Field.objects.all()
34      serializer_class = FieldSerializer
35
36
37  class PointOfInterestViewSet(viewsets.ModelViewSet):
38      queryset = PointOfInterest.objects.all()
39      serializer_class = PointOfInterestSerializer
40
41
42  class CultivationViewSet(viewsets.ModelViewSet):
43      queryset = Cultivation.objects.all()
44      serializer_class = CultivationSerializer
45
```

Εικόνα 8. Ελεγκτές λογικής CRUD

Τέλος το DRF προσφέρει επίσης και μια διεπαφή χρήστη (UI) μέσω της οποίας προγραμματιστές αλλά και χρήστες μπορούν να κάνουν χρήση των λειτουργιών της διεπαφής με γραφικό τρόπο. Όπως φαίνεται στην εικόνα 9 η γραφική διεπαφή του API εμφανίζει με σαφή τρόπο τις διαθέσιμες ενέργειες που μπορούν εκτελεστούν σε έναν πόρο (κουμπιά δεξιά) όπως επίσης και το αποτέλεσμα εκτέλεσης ενός αιτήματος στην κέντρο.



Εικόνα 9. Γραφική διεπαφή API

4.2.2 Μεταφόρτωση αρχείων χωρικών δεδομένων

Εκτός από την βασική λειτουργικότητα (CRUD) που μόλις περιγράψαμε υπάρχει η δυνατότητα εκτέλεσης ειδικού κώδικα (custom) για συγκεκριμένες περιπτώσεις όπως για παράδειγμα κατά την είσοδο μαζικών δεδομένων με την μεταφόρτωση ενός αρχείου (shapefile bundle).

Στην εικόνα 10 παρατίθεται ο ειδικός κώδικας για την αποδοχή του αρχείου, την αποσυμπίεση του και την κλήση ειδικής μεθόδου (*γραμμή 125 – «run»*) για την διαχείριση των δεδομένων

```
94
95  class FileUploadView(views.APIView):
96
97      def put(self, request, filename, format=None):
98          # Save uploaded file
99          file_obj = self.request.data['file']
100         path = default_storage.save(
101             '/tmp/temp.zip', ContentFile(file_obj.read()))
102         model = self.request.query_params.get('model', None)
103
104         # Decompress file
105         try:
106             with zipfile.ZipFile(path, 'r') as zip_ref:
107                 member_list = zip_ref.filelist
108                 zip_ref.extractall('/tmp')
109         except Exception as error:
110             print('\n***** Exception being handled *****\n')
111             print('Decompressing file failed: ', error)
112
113             return Response(status=500, data={'message': 'Decompressing zip file failed!'})
114
115         # Fetch shapefile
116         shapefile = None
117         for member in member_list:
118             filename = member.filename
119             if filename.endswith('.shp'):
120                 shapefile = filename
121
122         bad_input_error_flag = False
123         error_reason = ''
124         try:
125             run(model, '/tmp/' + shapefile, True)
126         except Exception as error:
127             print('\n***** Exception being handled *****\n')
128             print('Error occured decoding the file!', error)
129             bad_input_error_flag = True
130             error_reason = str(error)
131
132         # Clean up storage
133         default_storage.delete(path)
134
135         folder = '/tmp'
136         for filename in os.listdir(folder):
137             file_path = os.path.join(folder, filename)
138             try:
139                 if os.path.isfile(file_path) or os.path.islink(file_path):
140                     if filename in member_list:
141                         os.unlink(file_path)
142                 elif os.path.isdir(file_path):
143                     shutil.rmtree(file_path)
144             except Exception as e:
145                 print('Failed to delete %s. Reason: %s' % (file_path, e))
146         if(bad_input_error_flag):
147             print('ERROR: ', error_reason)
148
149             return Response(status=400, data={'message': 'Bad Input', 'error': error_reason})
150
151         return Response(status=201)
```

Εικόνα 10. Ελεγκτής λογικής μεταφόρτωσης αρχείου

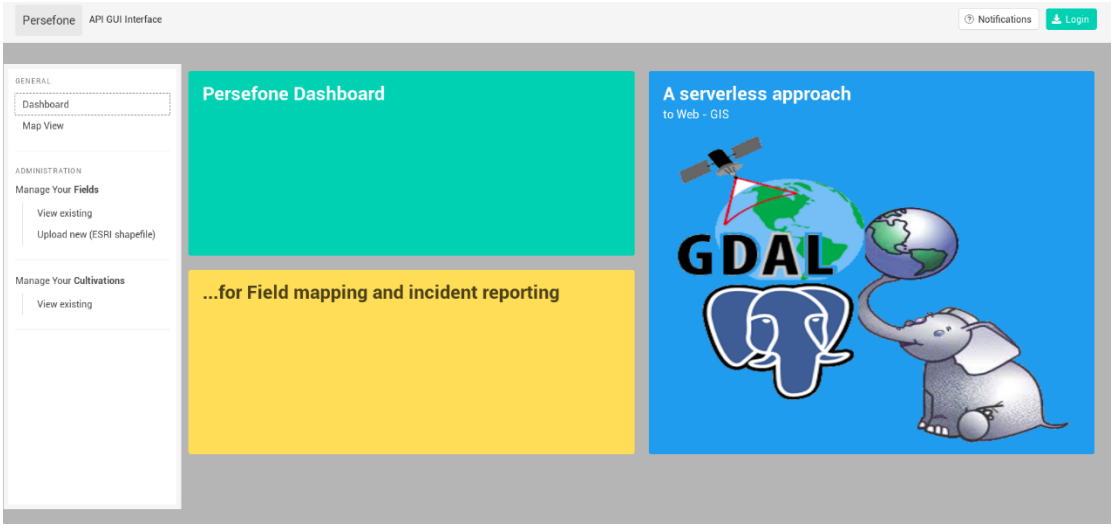
Η μέθοδος με την ονομασία “run” είναι ένα πολύ μικρό τμήμα κώδικα το οποίο χρησιμοποιώντας τις διεπαφές για την βιβλιοθήκη *GDAL* διαβάζει και καταχωρεί τα δεδομένα, τόσο χωρικά όσο και περιγραφικά, από τα κατάλληλα αρχεία (.shp, .shx και .dbf) στην βάση δεδομένων. Στην εικόνα 10 φαίνεται ο κώδικας της μεθόδου. Πιο συγκεκριμένα η λειτουργικότητα της ανάγνωσης των δεδομένων και της αποθήκευσης αυτών γίνεται από την κλάση LayerMapping η οποία στην ουσία αποτελεί ένα υψηλού επιπέδου εργαλείο για την εισαγωγή δεδομένων από αρχεία τύπου shapefile κάνοντας κλήσεις στις μεθόδους και κλάσεις της *GDAL / OGR* βιβλιοθήκης.

```
field_mapping = {
    'name': 'Name',
    'location': 'MULTIPOLYGON',
}

cultivation_mapping = {
    'name': 'Name',
    'location': 'MULTIPOLYGON',
    'field': {'id': 'Field'},
}

poi_mapping = {
    'name': 'Name',
    'location': 'POINT',
    'humidity': 'Humidity',
    'temperature': 'Temp',
    'luminocity': 'Luminocity',
    'field': {'id': 'Field'}
}


def run(model, shape, verbose=True):
    print('Running..')
    print('LayerMapping instanciation..')

    if model == 'Field':
        model = Field
        mapping = field_mapping
    elif model == 'Cultivation':
        model = Cultivation
        mapping = cultivation_mapping
    elif model == 'PointOfInterest':
        model = PointOfInterest
        mapping = poi_mapping
    else:
        raise ValueError('Invalid model case!')

    lm = LayerMapping(model, shape, mapping, transform=False)

    print('Save operation..')
    lm.save(strict=True, verbose=verbose)
```

Εικόνα 11. Δημιουργία & αποθήκευση γεωχωρικών δεδομένων από αρχείο

4.3 Γραφική διασύνδεση

Όπως ήδη αναφέραμε στο προηγούμενο κεφάλαιο η υλοποίηση του πιλοτικού προγράμματος πελάτη (γραφική διασύνδεση χρήστη) πραγματοποιήθηκε με την χρήση της React. Για να κάνουμε ευκολότερη και πρακτικότερη την ανάπτυξη επιλέξαμε η φιλοξενία του πελάτη να γίνει και αυτή από την υπηρεσία της Lambda. Πριν προχωρήσουμε στην περιγραφή του γραφικής διεπαφής χρήστη να σημειώσουμε ότι οποιοδήποτε πρόγραμμα πελάτης θα μπορούσε να χρησιμοποιηθεί και επιπλέον ότι το πρόγραμμα θα μπορούσε να φιλοξενείται οπουδήποτε αφού η λογική της εφαρμογής είναι πάντα προσβάσιμη μέσω της διεπαφής ιστού που μόλις περιγράψαμε και υποστηρίζει όλες τις πλατφόρμες.

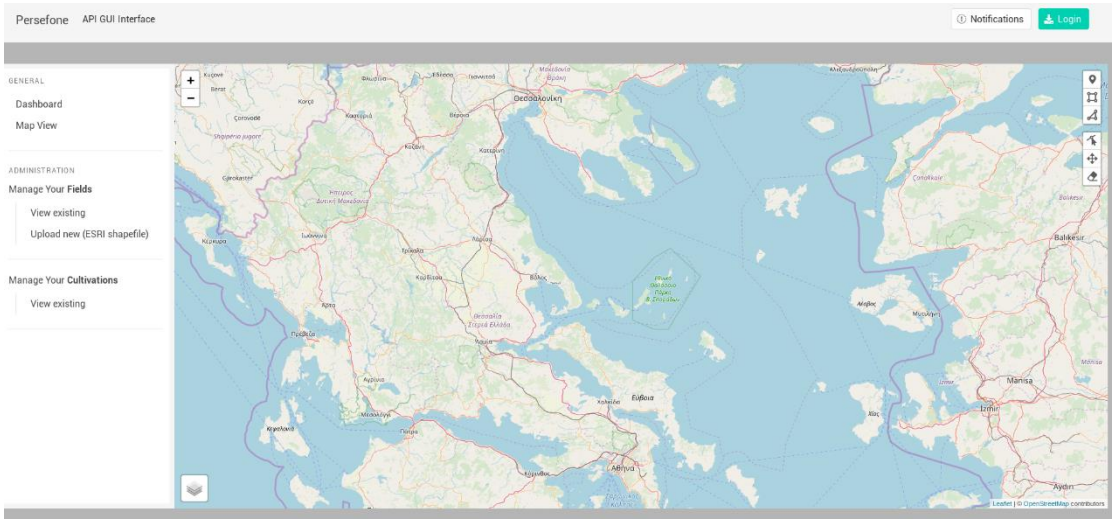


Εικόνα 12. Αρχική εικόνα γραφικής διασύνδεσης

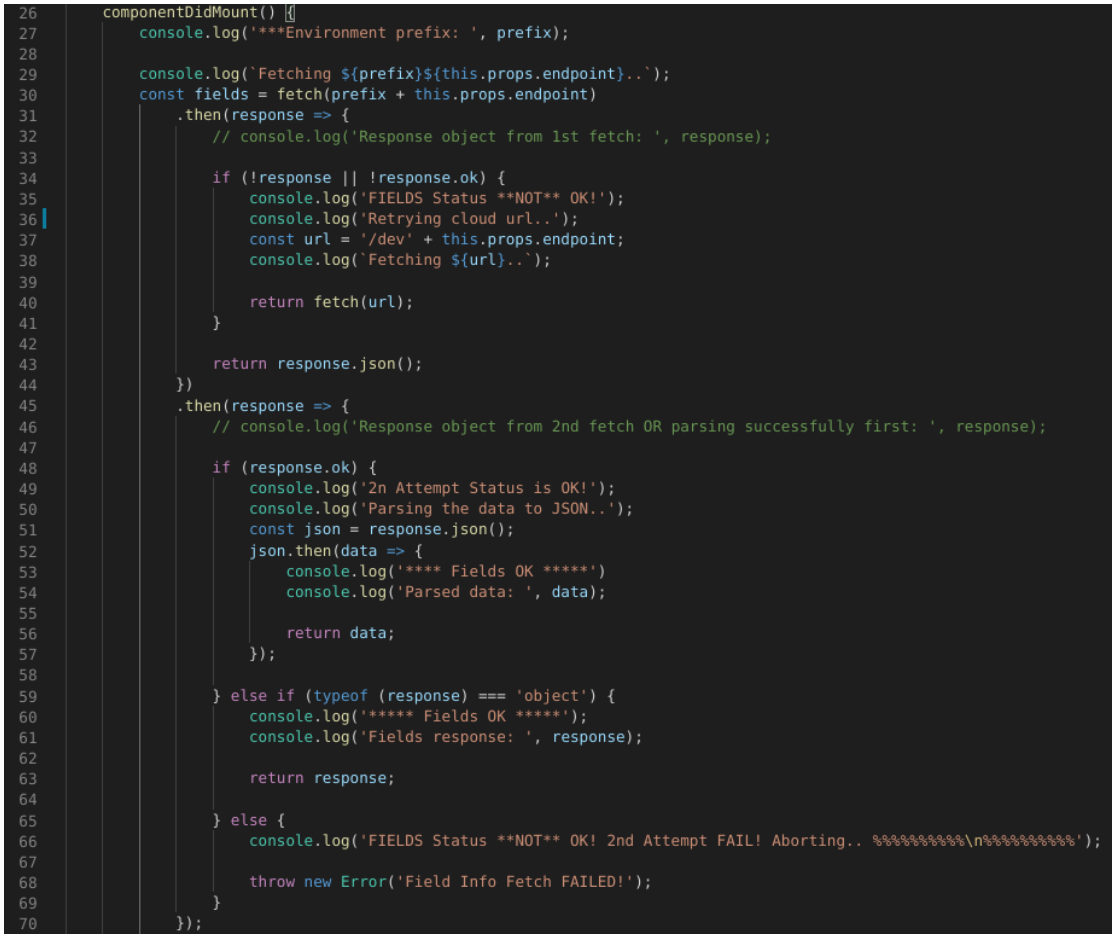
Το βασικό URL (root) της εφαρμογής είναι ρυθμισμένο ώστε να σερβίρει μια σελίδα HTML (index.html) στην οποία είναι ενσωματωμένος ο κώδικας της React εφαρμογής στην οποία είναι υλοποιημένη η γραφική διασύνδεση χρήστη. Η αρχική σελίδα εικονίζεται παραπάνω και όπως βλέπουμε αποτελείται από μια μπάρα βασικής πλοήγησης, ένα πλαϊνό παράθυρο πλοήγησης λειτουργιών και την κεντρική θέαση. Η μπάρα πλοήγησης χρησιμεύει απλώς για την επαναφορά στην αρχική σελίδα ή την μετάβαση στην γραφική διεπαφή του API (βλέπε προηγούμενη παράγραφο). Η λειτουργικότητα της εφαρμογής ελέγχεται από το πλαϊνό πλαίσιο μέσω του οποίου ο χρήστης μπορεί να επιλέξει ανάμεσα σε τρεις διαφορετικές θεάσεις:

1) Web Map για ανάκτηση και εισαγωγή δεδομένων
2) Παρουσίαση υπαρχόντων δεδομένων (σε κείμενο / λίστα)
3) Εισαγωγή δεδομένων με αρχείο (shapefile)

Επιλέγοντας την υπερσύνδεση Web Map η κεντρική θέαση αλλάζει μετά από κάποια δευτερόλεπτα σε έναν διαδραστικό χάρτη όπως αυτός της εικόνας 13. Κατά την φόρτωση του χάρτη αποστέλλονται HTTP αιτήματα (ένα για κάθε μοντέλο) προς την διεπαφή της εφαρμογής με σκοπό την ανάκτηση των αποθηκευμένων δεδομένων και την απεικόνιση τους στο χάρτη. Η υλοποίηση του μηχανισμού των αιτημάτων παρατίθεται στις εικόνες 14 και 15. Πρόκειται όπως βλέπουμε για τρεις ασύγχρονες κλήσης του fetch API του φυλλομετρητή αποφεύγοντας έτσι εγκατάσταση επιπλέον βιβλιοθηκών. Εάν παρατηρήσουμε με μεγαλύτερη λεπτομέρεια η υλοποίηση περιλαμβάνει και έναν μηχανισμό επανάκλησης για το ενδεχόμενο μια Lambda συνάρτηση αργήσει να αποκριθεί (βλέπε παρ. 3.1 – AWS Lambda). Μετά την απόκριση των αιτημάτων τα δεδομένα μετατρέπονται σε επίπεδα με την βοήθεια της βιβλιοθήκης react-leaflet και εμφανίζονται στον χάρτη. Η προαναφερθείσα βιβλιοθήκη προσφέρει έτοιμα React στοιχεία (components) GeoJSON επίπεδα σε έναν Leaflet χάρτη εξοικονομώντας έτσι γραμμές κώδικα. Ο χρήστης έχει επίσης την δυνατότητα απόκρυψης και εμφάνισης των διαφορετικών επιπέδων ανάλογα με τις προτιμήσεις του χρήστη.



Εικόνα 13. Θέαση χάρτη γραφικής διασύνδεσης



```
componentDidMount() {
    console.log('***Environment prefix: ', prefix);

    console.log(`Fetching ${prefix}${this.props.endpoint}..`);
    const fields = fetch(prefix + this.props.endpoint)
        .then(response => {
            // console.log('Response object from 1st fetch: ', response);

            if (!response || !response.ok) {
                console.log('FIELDS Status **NOT** OK!');
                console.log('Retrying cloud url..');
                const url = '/dev' + this.props.endpoint;
                console.log(`Fetching ${url}..`);

                return fetch(url);
            }

            return response.json();
        })
        .then(response => {
            // console.log('Response object from 2nd fetch OR parsing successfully first: ', response);

            if (response.ok) {
                console.log('2n Attempt Status is OK!');
                console.log('Parsing the data to JSON..');
                const json = response.json();
                json.then(data => {
                    console.log('**** Fields OK *****')
                    console.log('Parsed data: ', data);

                    return data;
                });

            } else if (typeof (response) === 'object') {
                console.log('***** Fields OK *****');
                console.log('Fields response: ', response);

                return response;

            } else {
                console.log('FIELDS Status **NOT** OK! 2nd Attempt FAIL! Aborting.. %%%%%%%%%%\n%%%%%%%%%%');

                throw new Error('Field Info Fetch FAILED!');
            }
        });
```

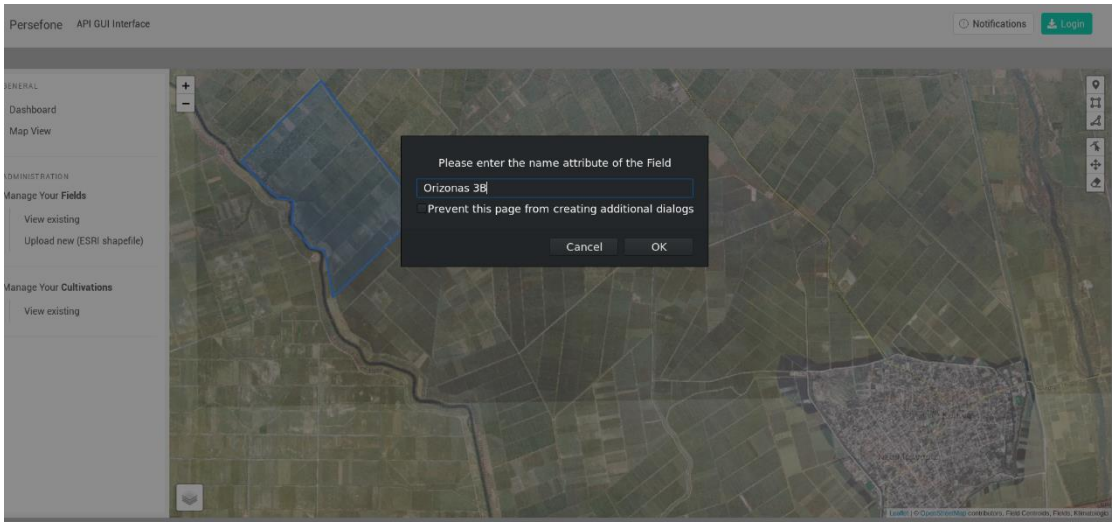
Εικόνα 14. Αιτήματα ανάκτησης δεδομένων (1)

```
    Promise.all([fields, centroids, points, cultivations])
        .then(response => {
            console.log('DataProvider fetched fine..');

            this.setState({
                data: response[0],
                data2: response[1],
                points: response[2],
                cultivations: response[3],
                loaded: true
            });
        })
        .catch(error => {
            console.log('Error in DataProvider:', error);

            this.setState({
                loaded: false
            });
        });
}

render() {
    console.log('DataProvider rendered');
    const href = window.location.href;
    if (href.includes('localhost') || href.includes('192.168')) {
        prefix = '';
    }

    const { data, data2, cultivations, points, loaded, placeholder } = this.state;
    return loaded ? this.props.render(data, data2, points, cultivations) : <div style={{ margin: '0 auto', verticalAlign: 'middle' }}>{placeholder}</div>;
}
}
```

Εικόνα 15. Αιτήματα ανάκτησης δεδομένων (2)

Για την εισαγωγή των δεομένων ο χρήστης έχει την δυνατότητα να σχεδιάσει τα πολύγωνα που τον ενδιαφέρουν όπως και σημεία ενδιαφέροντος ή μετρήσεις. Με την σχεδίαση ενός πολυγώνου ή σημείου εμφανίζεται στον χρήστη παράθυρο διαλόγου να (εικόνα 16) προκειμένου ο πρώτος να εισάγει τα περιγραφικά δεδομένα σχετικά με το

διάνυσμα που σχεδίασε. Έπειτα τα δεδομένα δομούνται κατάλληλα (GeoJSON) και αποστέλλονται στην διεπαφή.



Εικόνα 16. Θέαση χάρτη - Εισαγωγή δεδομένων γραφικής διασύνδεσης

Τέλος, στη τρίτη θέαση ο χρήστης έχει την δυνατότητα να εισάγει δεδομένα μαζικά με το ανέβασμα κατάλληλου αρχείου (zip). Με την πλοήγηση στην αντίστοιχη ενότητα (upload new) ο χρήστης δύναται να αποστείλει ένα συμπιεσμένο αρχείο το οποίο θα περιλαμβάνει τα απαραίτητα αρχεία γεωχωρικής (.shp, .shx) και περιγραφικής πληροφορίας (.dbf).

5. Συμπεράσματα & προτάσεις

Στο κεφάλαιο αυτό παρουσιάζονται τα συμπεράσματα τα οποία προέκυψαν από την αρχική έρευνα και μελέτη που έγινε, μέχρι τον τελικό σχεδιασμό και υλοποίηση του πρωτοτύπου. Επίσης, αναφέρονται κάποιες προτάσεις για μελλοντικές βελτιώσεις και διατάξεις του πρωτοτύπου.

5.1. Συμπεράσματα

Με την παρούσα διπλωματική εργασία επιχειρήσαμε να εξετάσουμε και να απαντήσουμε στο ερευνητικό ερώτημα που σχετίζεται με το αν είναι εφικτό και πρακτικό να χρησιμοποιήσουμε Cloud τεχνολογίες και ειδικότερα αυτή της Υπολογιστικής άνευ Διακομιστών σε συνδυασμό με τις τεχνολογίας των Γεωγραφικών Πληροφοριακών Συστημάτων. Αρχικά εξετάσαμε και αναλύσαμε τα χαρακτηριστικά των προαναφερθέντων τεχνολογιών. Έπειτα, περιγράψαμε την αρχιτεκτονική και τα εργαλεία για την υλοποίηση του πρωτοτύπου και κάναμε μια συνοπτική αναφορά στις διαθέσιμες εναλλακτικές. Τέλος δημιουργήσαμε μια πρωτόλεια πιλοτική Serverless GIS εφαρμογή σε μια πρώτη απόπειρα εμπειρικής επαλήθευσης της πρότερης θεωρητικής μας κατασκευής. Η λειτουργία της έδειξε ότι η θεωρητική σύνθεση, οι βασικές ερευνητικές υποθέσεις και η βασική μας αντίληψη σχετικά με την εγκυρότητα και επικαιρότητα των συγκεκριμένων τεχνολογιών εμφανίζει να έχει πρακτική τεχνολογική εφαρμογή.

5.2. Μελλοντική Έρευνα

- Δημιουργία/ δοκιμή κινητής εφαρμογής για την συλλογή δεδομένων στο πεδίο και επικοινωνία με την εφαρμογή ιστού
- Δοκιμή γεωχωρικών υπολογισμών υψηλών απαιτήσεων όπως εκτέλεση IDW
- Stress testing για Performance analysis
- Επέκταση της βιβλιοθήκης Zappa για την υποστήριξη της μεταφοράς GeoDjango εφαρμογών σε άλλους Cloud Vendors αλλά και σε Open Cloud υλοποιήσεις

Βιβλιογραφία

Ξένη και Μεταφρασμένη

Adzic, G., & Chatley, R. (2017). Serverless computing: economic and architectural impact. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering - ESEC/FSE 2017* (pp. 884–889). New York, New York, USA: ACM Press. https://doi.org/10.1145/3106237.3117767

Agarwal, M., & Saran Srivastava, G. M. (2017). Cloud Computing: A Paradigm Shift in the Way of Computing. *International Journal of Modern Education and Computer Science*, *9*(12), 38–48. https://doi.org/10.5815/ijmecs.2017.12.05

Ahmad Dar, A. (2018). Cloud Computing-Positive Impacts and Challenges in Business Perspective. *Journal of Computer Science & Systems Biology*, *12*(01), 15–18. https://doi.org/10.4172/jcsb.1000294

Castro, P., Ishakian, V., Muthusamy, V., & Slominski, A. (2019). The rise of serverless computing. *Communications of the ACM*, *62*(12), 44–54. https://doi.org/10.1145/3368454

Chen, T., Ta-Tao, C., & Kazuo, N. (2016). The Perceived Business Benefit of Cloud Computing: An Exploratory Study. *Journal of International Technology & Information Management*, *25*(4), 101–121. Retrieved from http://0-search.ebscohost.com.lib1000.dlsu.edu.ph/login.aspx?direct=true&db=bth&AN=122416572&site=eds-live

Hogan, M., Liu, F., Sokol, A., & Tong, J. (2011). NIST Cloud Computing Standards Roadmap. *National Institure of Standards and Technology*.

McCarthy, J. (1992). Reminiscences on the History of Time-Sharing. *IEEE Annals of the History of Computing*, *14*(1), 19–24.

Pérez, A., Moltó, G., Caballer, M., & Calatrava, A. (2018). Serverless computing for container-based architectures. *Future Generation Computer Systems*, *83*, 50–59. https://doi.org/10.1016/j.future.2018.01.022

Sehgal, N. K., & Bhatt, P. C. P. (2018). *Cloud Computing* (Vol. 17). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-77839-6

Singh, S. (2019). Cloud Computing : An Overview. *International Journal of Scientific Research in Science, Engineering and Technology*, *7*(4), 63–70. https://doi.org/10.32628/IJSRSET196120

Soldani, J., Tamburri, D. A., & Van Den Heuvel, W. J. (2018). The pains and gains of microservices: A Systematic grey literature review. *Journal of Systems and Software*, *146*, 215–232. https://doi.org/10.1016/j.jss.2018.09.082

Varghese, B. (2019). A History of the Cloud. *ITNOW*, *61*(2), 46–48. https://doi.org/10.1093/itnow/bwz049

Wieczorek, W. F., & Delmerico, A. M. (2009). Geographic information systems. *Wiley Interdisciplinary Reviews: Computational Statistics*, *1*(2), 167–186. https://doi.org/10.1002/wics.21

Xue, C. T. S., & Xin, F. T. W. (2016). Benefits and Challenges of the Adoption of Cloud Computing in Business. *International Journal on Cloud Computing: Services and Architecture*, *6*(6), 01–15. https://doi.org/10.5121/ijccsa.2016.6601

Yussupov, V., Breitenbücher, U., Leymann, F., & Wurster, M. (2019). A Systematic Mapping Study on Engineering Function-as-a-Service Platforms and Tools. In *Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing* (pp. 229–240). New York, NY, USA: ACM. https://doi.org/10.1145/3344341.3368803

Διαδίκτυο

https://www.ibm.com/cloud/learn/cloud-computing

https://goizueta.emory.edu/faculty/profiles/ramnath-k-chellappa

https://www.gartner.com/en/newsroom/press-releases/2019-04-02-gartner-forecasts-worldwide-public-cloud-revenue-to-g

https://pages.awscloud.com/Gartner-Magic-Quadrant-for-Infrastructure-as-a-Service-Worldwide.html?trk=ar_carousel

https://medium.com/@jaychapel/alibaba-cloud-market-share-2019-25d30bc096f7

https://medium.com/cloudpegboard/how-many-aws-services-are-there-51dda44fa946

https://www.2ndwatch.com/blog/popular-aws-products-2018/

https://www.zdnet.com/article/aws-big-data-machine-learning-services-gain-traction-says-2nd-watch/

https://www.zdnet.com/article/top-cloud-providers-2019-aws-microsoft-azure-google-cloud-ibm-makes-hybrid-move-salesforce-dominates-saas/

https://gdpr-info.eu/art-33-gdpr/

https://www.esri.com/

https://esripress.esri.com/storage/esripress/images/188/115391_webgis_chapter01.pdf

https://www.opengeospatial.org/standards/wms

https://www.opengeospatial.org/standards/wps

https://www.opengeospatial.org/standards/wfs

https://www.opengeospatial.org/standards/gml

https://www.opengeospatial.org/standards/eo-geojson

https://www.postgresql.org/

https://www.python.org/

https://brochure.getpython.info/

https://en.wikipedia.org/wiki/History_of_Python

https://en.wikipedia.org/wiki/Python_%28programming_language%29

https://www.djangoproject.com/

https://wiki.python.org/moin/WebFrameworks/

https://docs.djangoproject.com/en/3.0/ref/contrib/gis/

https://trac.osgeo.org/geos/

https://gdal.org/

https://proj.org/

https://github.com/Miserlou/Zappa

https://www.django-rest-framework.org/

https://reactjs.org/

https://leafletjs.com/

https://agafonkin.com/

https://react-leaflet.js.org/

https://github.com/geoman-io/leaflet-geoman

https://github.com/openwisp/django-rest-framework-gis