ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ
ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

AI-Driven Marathon Analytics and Runner Insights Platform Utilizing AWS Services

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Χρήστος Λάσδας

Επιβλέπων : Παναγιώτης Τσανάκας
Καθηγητής Ε.Μ.Π

Αθήνα, Ιούλιος 2025

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ
ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

AI-Driven Marathon Analytics and Runner Insights Platform Utilizing AWS Services

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Χρήστος Λάσδας

Επιβλέπων : Παναγιώτης Τσανάκας
Καθηγητής Ε.Μ.Π

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 12η Ιουλίου 2025

……..	……..	……..
Παναγιώτης Τσανάκας	Γεώργιος Ματσόπουλος	Ανδρέας-Γεώργιος Σταφυλοπάτης
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2025

.....

Χρήστος Λάσδας

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Ευχαριστίες

Η διπλωματική αυτή εκπονήθηκε για την σχολή των Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Εθνικού Μετσόβιου Πολυτεχνείου, υπό την επίβλεψη του Καθηγητή Παναγιώτη Τσανάκα.

Θα ήθελα να ευχαριστήσω θερμά τον κοσμήτορα κ. Παναγιώτη Τσανάκα για την υποστήριξη και καθοδήγηση που μου προσέφερε, καθώς και για την ευκαιρία που μου έδωσε να ασχοληθώ με ένα τέτοιο θέμα.

Ιδιαίτερες ευχαριστίες απευθύνονται στον υποψήφιο διδάκτορα Βρεττό Μουλό, ο οποίος με βοήθησε καθ' όλη τη διάρκεια της διπλωματικής με τις καινοτόμες ιδέες του και τις εύστοχες παρατηρήσεις συμβάλλοντας καθοριστικά στο τελικό αποτέλεσμα, καθώς και στον συνεργάτη του Δημήτρη Λιάπη για την παραχώρηση των αναγκαίων δεδομένων. Ακόμα στο GRNET , χάρη στο οποίο είχα πρόσβαση στις υπηρεσίες της Amazon.

Τέλος το μεγαλύτερο ευχαριστώ πάει στην οικογένεια μου, για την στήριξη τους και την βοήθεια τους καθόλη τη διάρκεια της διπλωματικής μου, των σπουδών μου, αλλά και της ζωής μου γενικότερα.

Αθήνα, Ιούλιος 2025
Χρήστος Λάσδας

Περίληψη

Στους αγώνες δρόμου που διοργανώνονται από διάφορους φορείς, μετά την πραγματοποίηση τους, οι διοργανωτές καταλήγουν με ένα μεγάλο σύνολο φωτογραφιών οι οποίες πρέπει να αντιστοιχηθούν σε κάθε δρομέα, διαδικασία εξαιρετικά χρονοβόρα αν γίνει χειροκίνητα, ώστε ο κάθε ένας να μπορεί να βλέπει εύκολα αυτές που τον ενδιαφέρουν. Ακόμα σε τέτοιους αγώνες συλλέγονται πολλές πληροφορίες σχετικά με την επίδοση του δρομέα στον αγώνα αλλά και τον ίδιο τον δρομέα, τις οποίες ο δρομέας επιθυμεί να δει μετά τη λήξη της διοργάνωσης. Συνεπώς είναι ιδιαίτερα χρήσιμη η ύπαρξη ενός αποδοτικού τρόπου κατηγοριοποίησης των εικόνων, αλλά και η ύπαρξη μιας εφαρμογής, στην οποία ο χρήστης να μπορεί να δει τα αποτελέσματα. Έτσι, στα πλαίσια αυτής της διπλωματικής, ασχολούμαι με την αυτόματη αντιστοίχιση των δρομέων στις φωτογραφίες τους, χρησιμοποιώντας τεχνικές τεχνητής νοημοσύνης , αλλά και με την ανάπτυξη μιας φιλικής προς τον χρήστη, serverless, web εφαρμογής, με χρήση των υπηρεσιών της Amazon.

Λέξεις Κλειδιά:

Αγώνας δρόμου, Αναγνώριση εικόνας, OCR, Face Recognition, InsightFace, AWS, DynamoDB, S3 object storage, Amazon Lambda, Web Service, REST API, Amazon Bedrock, Next.js, React, AntD, Python, Typescript, Serverless Architecture.

Abstract

In road races organized by various entities, a large number of photos are captured, which must be matched to the respective runners after the event. This process is extremely time-consuming when done manually, yet essential so that each participant can easily access the photos that concern them. Additionally, such events generate a wealth of information related to each runner's performance and personal data, which runners typically wish to review once the race is over. Therefore, an efficient method for categorizing these images, along with an application through which users can view the results, is highly valuable. In this thesis, I focus on the automatic matching of runners to their corresponding race photos using artificial intelligence techniques, as well as the development of a user-friendly, serverless web application built with Amazon Web Services.

Keywords:

Road race, Image recognition, OCR, Face Recognition, InsightFace, AWS, DynamoDB, S3 object storage, Amazon Lambda, Web Service, REST API, Amazon Bedrock, Next.js, React, AntD, Python, Typescript, Serverless Architecture.

Περιεχόμενα

Table of Contents

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Εισαγωγή

Τα τελευταία χρόνια το τρέξιμο έχει εξελιχθεί σε μία από τις δημοφιλέστερες μορφές άσκησης, αυτό έχει οδηγήσει στην διεξαγωγή διαφόρων αγώνων, από τους δήμους ή και άλλους φορείς, για τους φίλους του αθλήματος. Στους αγώνες αυτούς οι διοργανωτές, μεταξύ άλλων, έχουν μεριμνήσει για την χρονομέτρηση των δρομέων αλλά και για την φωτογραφική κάλυψη του αγώνα, ώστε ο κάθε συμμετέχων να μπορεί, μετά το τέλος του αγώνα, να βλέπει την επίδοση του και τις φωτογραφίες του αγώνα. Ωστόσο, προκύπτει το πρόβλημα της αντιστοίχισης των φωτογραφιών με τους αντίστοιχους δρομείς — μια διαδικασία εξαιρετικά χρονοβόρα όταν πραγματοποιείται χειροκίνητα.

Παράλληλα, η τεχνητή νοημοσύνη έχει πραγματοποιήσει μία πρωτοφανή και αξιοθαύμαστη πρόοδο, σε διάφορους τομείς, από την πραγματοποίηση προβλέψεων με βάση μία αριθμητική ακολουθία, μέχρι το Generative AI για την δημιουργία φυσικής γλώσσας και εικόνων/βίντεο. Αυτή η τεχνολογία έχει εξαπλωθεί σε τέτοιο βαθμό που πλέον είναι παρών στην καθημερινότητα μας, με βασικό παράδειγμα τα μοντέλα LLM τα οποία έχουν εξελιχθεί σε προσωπικούς βοηθούς μας. Ακόμη τα μοντέλα ανάλυσης εικόνων έχουν αναπτυχθεί και καταφέρνουν πολύ καλή, παρόμοια με την ανθρώπινη, απόδοση σε διεργασίες όπως ο εντοπισμός και η αναγνώριση κειμένου ή προσώπων σε μία εικόνα.

Ακόμα, η τεχνολογία του cloud έχει εξελιχθεί σε έναν από τους πιο καθοριστικούς παράγοντες για τον ψηφιακό μετασχηματισμό επιχειρήσεων και οργανισμών παγκοσμίως. Το cloud computing, δηλαδή η αποθήκευση, η επεξεργασία και η διαχείριση δεδομένων και εφαρμογών μέσω του διαδικτύου, προσφέρει ευελιξία, ταχύτητα και σημαντική μείωση κόστους. Η δημοτικότητα του cloud συνεχίζει να αυξάνεται, καθώς όλο και περισσότερες εταιρείες μεταφέρουν τις υποδομές τους σε αυτό, αναζητώντας μεγαλύτερη ασφάλεια, επεκτασιμότητα και καινοτομία.

Λαμβάνοντας υπόψη τα παραπάνω, στην παρούσα διπλωματική εργασία διερευνώ τη χρήση τεχνικών τεχνητής νοημοσύνης για την αυτόματη κατηγοριοποίηση φωτογραφιών από αγώνες δρόμου, αντιστοιχίζοντας κάθε εικόνα στον σωστό δρομέα. Με αυτόν τον τρόπο η διαδικασία του labeling των εικόνων παύει να είναι χειροκίνητη αλλά γίνεται αυτοματοποιημένα, εξοικονομώντας έτσι πολύτιμο χρόνο για τους διοργανωτές τέτοιων εκδηλώσεων. Παράλληλα, αναπτύσσω εφαρμογή φιλική προς τον χρήστη, μέσω της οποίας παρουσιάζονται τα αποτελέσματα και οι σχετικές εικόνες, αξιοποιώντας τις υπηρεσίες cloud της Amazon, του πιο δημοφιλούς παρόχου τέτοιων υπηρεσιών.

Κεφάλαιο 1 Ανάλυση Αλγορίθμων Τεχνητής Νοημοσύνης

1.1 Μοντέλα κειμένου

Τα LLM (Large Language Models) είναι προηγμένα μοντέλα τεχνητής νοημοσύνης που έχουν εκπαιδευτεί σε τεράστιες ποσότητες κειμένου με στόχο την κατανόηση και παραγωγή φυσικής γλώσσας. Μπορούν να απαντούν σε ερωτήσεις, να συνοψίζουν κείμενα, να γράφουν περιεχόμενο, να μεταφράζουν γλώσσες και να εκτελούν σύνθετες λογικές και δημιουργικές εργασίες. Χάρη στη βαθιά εκπαίδευσή τους, τα LLM αποτελούν τη βάση πολλών σύγχρονων εφαρμογών τεχνητής νοημοσύνης.

Στα πλαίσια της εργασίας αποφάσισα να χρησιμοποιήσω κάποιο LLM ώστε να δημιουργώ ένα προσωποποιημένο σχόλιο πάνω στην απόδοση του δρομέα και να γίνεται πιο φιλική η εμπειρία χρήσης. Για τον σκοπό αυτό, επέλεξα να χρησιμοποιήσω το Amazon Bedrock, το οποίο είναι μία serverless υπηρεσία της Amazon που επιτρέπει την on demand κλήση πληθώρας μοντέλων τεχνητής νοημοσύνης, λόγω της ευκολίας, των διαθέσιμων επιλογών αλλά και του ότι θα χρησιμοποιούσα και άλλες υπηρεσίες της Amazon κατά την ανάπτυξη της εφαρμογής. Παρακάτω λίγα λόγια για τα μοντέλα που δοκίμασα:

Claude 3.7 Sonnet

Το Claude 3.7 Sonnet της Anthropic αποτελεί ένα προηγμένο γλωσσικό μοντέλο που βασίζεται στην τεχνολογία «υβριδικής σκέψης» (hybrid reasoning), συνδυάζοντας τη δυνατότητα άμεσων αποκρίσεων με βαθύτερη, πολύπλοκη αναλυτική σκέψη μέσα στο ίδιο σύστημα. Διαθέτει δυναμική κλιμάκωση συμφραζομένων με παράθυρο έως και 200.000 tokens, επιτρέποντας την επεξεργασία μεγάλων κειμένων ή εγγράφων. Το μοντέλο αξιοποιεί τεχνικές αναστοχαστικού συλλογισμού (reflexive reasoning), έχει σχεδιαστεί με βάση την αρχή της «Συνταγματικής Τεχνητής Νοημοσύνης» (Constitutional AI) και εμφανίζει αυξημένη ακρίβεια, χαμηλά ποσοστά παραληρημάτων (hallucinations) και ασφάλεια στη χρήση. Είναι βελτιστοποιημένο για απαιτητικές εργασίες όπως λογική επίλυση προβλημάτων, προγραμματισμό και παραγωγή μη τοξικού περιεχομένου, ενώ υποστηρίζει λειτουργίες όπως function calling και RAG (Retrieval-Augmented Generation), καθιστώντας το κατάλληλο για σύνθετες επαγγελματικές και ερευνητικές εφαρμογές. [1]

Llama 3.2 3B

Το LLaMA 3.2 3B είναι ένα ελαφρύ αλλά ισχυρό μοντέλο φυσικής γλώσσας που αναπτύχθηκε από τη Meta ως μέρος της σειράς LLaMA (Large Language Model Meta AI). Με 3 δισεκατομμύρια παραμέτρους, αποτελεί μία από τις πιο αποδοτικές εκδοχές της οικογένειας LLaMA 3, προσφέροντας υψηλή ποιότητα γλωσσικής κατανόησης και παραγωγής με σημαντικά μειωμένες υπολογιστικές απαιτήσεις. Το μοντέλο έχει εκπαιδευτεί σε εκτεταμένα και ποιοτικά datasets, χρησιμοποιώντας τεχνικές fine-tuning και alignment, γεγονός που του επιτρέπει να επιτυγχάνει ανταγωνιστικές επιδόσεις σε διάφορα benchmarks. Είναι ιδανικό για ενσωμάτωση σε εφαρμογές χαμηλού κόστους, σε edge συσκευές ή ως μέρος serverless υποδομών, καθώς επιτυγχάνει εξαιρετική ισορροπία μεταξύ ταχύτητας, ακρίβειας και ενεργειακής απόδοσης. Χάρη στην ανοιχτή φιλοσοφία της Meta, το LLaMA 3.2 3B είναι διαθέσιμο σε προγραμματιστές και ερευνητές για ευρεία χρήση και πειραματισμό. [2]

Mistral Pixtral

Το Mistral Pixtral Large είναι ένα προηγμένο πολυτροπικό μοντέλο που αναπτύχθηκε από την εταιρεία Mistral, συνδυάζοντας επεξεργασία εικόνας και φυσικής γλώσσας σε ένα ενοποιημένο αρχιτεκτονικό πλαίσιο. Βασισμένο σε μοντέλο 124 δισεκατομμυρίων παραμέτρων, το Pixtral αξιοποιεί τεχνικές vision-language alignment ώστε να μπορεί να ερμηνεύει, να περιγράφει και να αναλύει οπτικά δεδομένα σε συνδυασμό με κείμενο. Υποστηρίζει μεγάλα παράθυρα συμφραζομένων, είναι ικανό να κατανοεί διαγράμματα, φωτογραφίες και γραφικές παραστάσεις, και ενδείκνυται για εφαρμογές όπως έγγραφα με ενσωματωμένες εικόνες, ανάλυση περιεχομένου ή ακόμη και ιατρική απεικόνιση. Το Pixtral Large είναι πλήρως εναρμονισμένο με τα υπόλοιπα μοντέλα της Mistral, με έμφαση στη διαφάνεια, την ταχύτητα και την αποδοτικότητα, ενώ διατίθεται μέσω API, σε cloud υποδομές όπως το Amazon Bedrock. [3]

Amazon Nova Pro

Το Amazon Nova Pro είναι ένα προηγμένο πολυτροπικό μοντέλο τεχνητής νοημοσύνης που αναπτύχθηκε από την Amazon Web Services και παρουσιάστηκε το 2024. Διαθέτει μεγάλο παράθυρο συμφραζομένων έως 300.000 tokens, υποστηρίζοντας την επεξεργασία κειμένου, εικόνων, βίντεο και άλλων πολυτροπικών δεδομένων. Το μοντέλο ξεχωρίζει για την υψηλή ακρίβεια και ταχύτητα, καθώς και για την υποστήριξη λειτουργικής κλήσης εξωτερικών εργαλείων, καθιστώντας το ιδανικό για σύνθετες εφαρμογές όπως ανάλυση πολυτροπικών δεδομένων, εκτέλεση πολύπλοκων εργασιών και ανάπτυξη εξατομικευμένων λύσεων. Το Nova Pro υποστηρίζει περισσότερες από 200 γλώσσες και επιτυγχάνει κορυφαίες επιδόσεις σε πολυάριθμα benchmarks, προσφέροντας παράλληλα ανταγωνιστικό κόστος σε σχέση με άλλα μοντέλα υψηλού επιπέδου. [4]

Amazon Titan G1 - Express

Το Amazon Titan G1 – Express είναι ένα μεγάλο γλωσσικό μοντέλο (LLM) που αναπτύχθηκε από την Amazon και έχει σχεδιαστεί για αποδοτική και ταχύτατη επεξεργασία φυσικής γλώσσας σε γενικού σκοπού εφαρμογές. Βασίζεται σε σύγχρονες τεχνικές μετασχηματιστών (transformers) και έχει βελτιστοποιηθεί για χαμηλή καθυστέρηση, υποστηρίζοντας λειτουργίες όπως δημιουργία και σύνοψη κειμένου, εξαγωγή πληροφοριών, συνομιλιακή αλληλεπίδραση, καθώς και ενσωμάτωση σε μηχανισμούς RAG (Retrieval-Augmented Generation). Διαθέτει παράθυρο συμφραζομένων έως 8.000 tokens, επιτρέποντας τη διαχείριση εκτεταμένων εισόδων και διαλόγων, ενώ υποστηρίζει περισσότερες από 100 γλώσσες (σε preview). Η αρχιτεκτονική του μοντέλου έχει βελτιστοποιηθεί για ισορροπία μεταξύ ακρίβειας, ταχύτητας και κόστους, ενώ περιλαμβάνει δυνατότητες όπως function calling, έλεγχο της δημιουργικότητας μέσω παραμέτρων (temperature, top-p) και υποστήριξη ροής (streaming) για αποκρίσεις σε πραγματικό χρόνο. Χάρη στον αποδοτικό του σχεδιασμό, το Titan G1 – Express είναι ιδιαίτερα κατάλληλο για παραγωγικές εφαρμογές σε περιβάλλοντα που απαιτούν υψηλή απόδοση με περιορισμένους πόρους. [5]

Δοκίμασα τα παραπάνω μοντέλα μέσω του Bedrock API , ζητώντας τους να μου δώσουν ένα σύντομο σχόλιο για την απόδοση ενός δρομέα , δίνοντας τους τα δεδομένα του σε μορφή json, παρακάτω φαίνονται τα αποτελέσματα μιας εκτέλεσης ενδεικτικά καθώς και το κόστος χρήσης κάθε μοντέλου:

Μοντέλο	**Παραγόμενο κείμενο**
Claude 3.7 Sonnet	Hey Vilho, way to dominate out there! First overall AND first in your age group? That's what I call Finnish-ing strong! 🇫🇮 Your time of 1:13:53 crushed your M25 age group's average by over 12 minutes. Those consistent splits show you weren't just racing—you were on a mission! IF Raseborg must be proud to have such a speedy Finn on their team. Congrats on the gold!
Llama 3.2 3B	``` Awesome job Vilho, you completed the 21km race and crushed it! I mean, who needs a personal record when you can set a national record? ``` This comment is a mix of encouragement and humor, implying that the runner did exceptionally well and that their performance should be celebrated. The mention of "national record" is a playful way to acknowledge their success, without making them feel like they need to improve further. Here is the code to generate this comment: ```python import pandas as pd data = { "rank": 1, "agegrouprank": 1, "genderrank": 1,

	"bib": 21276, "lastname": "*********", "firstname": "VILHO", "fathersname": "*********", "nationality": "FIN", "club":
Mistral Pixtral	Wow,Vilho, absolutely crushing it out there! Ranking 1st overall, 1st in your age group, and 1st among the guys—you're the king of the race! That 1:13:53 finish time is lightning fast, especially compared to the average time for your age group. Every split shows how consistently you dominated the course. Keep shining with IF RASEBORG—you're setting the bar high! 🎉🏃♂️
Amazon Nova Pro	Hey Vilho! 🏅🏃♂️ You absolutely crushed the 21km race! Not only did you finish first overall, but you also topped your age group and gender category. Your splits were on point, and you shaved off a solid 12 minutes from the mean time of your age group! 🕑🔥 Your club, IF Raseborg, must be super proud. Keep up the amazing work, and here's to many more victories! 🎉
Amazon Titan G1 - Express	Great job, Vilho ******! Your hard work and dedication have paid off in this 21km race, where you secured the top spot as the overall champion. Your impressive time of 01:13:53 demonstrates your exceptional endurance and speed. Your consistent splits throughout the race show that you maintained a strong pace, allowing you to finish with a remarkable time. Your age group's mean time of 01:26:16 also highlights your performance, as it demonstrates your ability to outperform your peers in your age category. Considering the number of participants in your age group, your achievement is even more significant. Overall, your victory in this 21km race is a testament to your talent, commitment, and determination. Congratulations on your well-deserved success!

Πίνακας 1: Ποιότητα εξόδου μοντέλων κειμένου

Μοντέλο	Χρόνος (sec)	Κόστος ανα 1000 input tokens ($)	Κοστος ανά 1000 output tokens ($)
Claude 3.7 Sonnet	4.94	0.003	0.015
Llama 3.2 3B	2.97	0.00019	0.00019
Mistral Pixtral	1.85	0.002	0.006
Amazon Nova Pro	1.63	0.00105	0.0042
Amazon Titan G1 - Express	9.11	0.0003	0.000863

Πίνακας 2: Κόστος και ταχύτητα μοντέλων κειμένου

Η συνάρτηση της python που καλεσα για κάθε μοντέλο:

```python
bedrock = boto3.client("bedrock-runtime", region_name="eu-central-1")

prompt_text = (
    "below is the information about the performance of a runner in a 21km race:\n"
    + json.dumps(info)
    + "\nmake a brief yet fun comment about their performance, considering the above info, as if you are talking to them.\n"
)

def invoke_bedrock_model(model_id, body_dict):
    start_time = time.time()
    response = bedrock.invoke_model(
        modelid=model_id,
        contenttype="application/json",
        accept="application/json",
        body=json.dumps(body_dict),
    )
    elapsed = time.time() - start_time
    response_body = response["body"].read()
    return json.loads(response_body), elapsed

```

Συμπεράσματα

Από τα παραπάνω μοντέλα που δοκίμασα, μετά από αρκετές εκτελέσεις,βλέπω ότι το κείμενο που παράγεται είναι αποδεκτό στις περιπτώσεις των Claude, Mistral , Nova Pro και Titan G1, το Llama παρόλο που διευκρινίζω στο prompt ότι θέλω μόνο το κείμενο απαντάει συμπεριλαμβάνοντας περιττό κείμενο που κάνει πιο απρόβλεπτη τη χρήση του.

Λαμβάνοντας υπόψην την ποιότητα της απάντησης, τον χρόνο ανταπόκρισης και το κόστος κάθε μοντέλου καταλήγω πως το ιδανικότερο μοντέλο για την χρήση που θέλω είναι το **Claude 3.7 Sonnet** . Αυτό επειδή δίνει πιο φιλικές, χιουμοριστικές απαντήσεις, ενώ παρόλο που είναι το δεύτερο πιο αργό, η καθυστέρηση των 3-5 δευτερολέπτων δεν αποτελεί πρόβλημα για την εμπειρία χρήσης, εφόσον δημιουργείται ένα ποιοτικό σχόλιο. Ακόμα το κόστος του μπορεί να είναι το πιο ακριβό από τα δοκιμασμένα μοντέλα, αλλά εξακολουθεί να είναι χαμηλό και δε θα αποτελέσει πρόβλημα για τον αριθμό των χρηστών που αναμένεται να έχει η εφαρμογή

1.2 Μοντέλα εικόνων

Τα μοντέλα ανάλυσης εικόνας είναι αλγοριθμικά συστήματα που χρησιμοποιούν τεχνικές τεχνητής νοημοσύνης, κυρίως deep learning, για να κατανοήσουν, ερμηνεύσουν και εξάγουν πληροφορίες από ψηφιακές εικόνες ή βίντεο. Βασίζονται κυρίως σε συνελικτικά νευρωνικά δίκτυα (CNNs) και χρησιμοποιούνται για ποικίλες εφαρμογές όπως αναγνώριση αντικειμένων, προσώπων και σκηνών, ανίχνευση δραστηριότητας, κατηγοριοποίηση εικόνων, εξαγωγή χαρακτηριστικών και επεξεργασία φυσικής γλώσσας με βάση το περιεχόμενο εικόνων. Τα σύγχρονα μοντέλα συνδυάζουν υψηλή ακρίβεια με δυνατότητες πραγματικού χρόνου, καθιστώντας τα ιδανικά για τομείς όπως η ασφάλεια, η ιατρική διάγνωση, τα αυτόνομα οχήματα και η βιομηχανική αυτοματοποίηση.

Στην τωρινή εφαρμογή θέλω να χρησιμοποιήσω τα μοντέλα αυτά για να βρω με καλή ακρίβεια ποιες φωτογραφίες, από όλο το σύνολο των φωτογραφιών, αντιστοιχούν σε κάθε δρομέα. Ψάχνοντας στο διαδίκτυο βρήκα προηγούμενες τεχνικές που έχουν δοκιμαστεί.

1.2.1 Προηγούμενες τεχνικές

Αναζητώντας πληροφορίες επί του θέματος στο διαδίκτυο βρήκα αρκετά papers που ασχολούνται με την αναγνώριση των bib numbers των δρομέων στις φωτογραφίες ενός αγώνα (bib number είναι ο αριθμός που φέρει κάθε δρομέας στη μπλούζα του) . Papers όπως τα , Racing Bib Number Recognition [6] , An Evaluation of General-Purpose Optical Character Recognizers and Digit Detectors for Race Bib Number Recognition [7], Marathon Bib Number Recognition using Deep Learning [8] και Marathon athletes number recognition model with compound deep neural network [9] ασχολούνται με τον εντοπισμό των περιοχών της εικόνας στις οποίες βρίσκονται τα bib numbers, και και μετά μέσω του Tesseract (ενός δημοφιλούς framework αναγνώρισης χαρακτήρων - OCR, optical character recognition) ή χρησιμοποιώντας convolutional recurrent neural networks (CRNN) αναγνωρίζουν το bib number. Ωστόσο αυτά τα papers περιορίζονται σε αναγνώριση του bib number, όχι σε tagging των φωτογραφιών με τα bib numbers των δρομέων που εμφανίζονται.

Ακόμα, χρήσιμο είναι ένα άρθρο στη πλατφόρμα Medium Marathon Bib Detection and Recognition [10] , στο οποίο αναφέρονται παρόμοιες τεχνικές εύρεσης του bib region και έπειτα αναγνώρισης του, ακόμα γίνεται λόγος και για τα πλεονεκτήματα μιας προσέγγισης με βάση την αναγνώριση προσώπου. Συγκεκριμένα αναφέρεται η μέθοδος στην οποία ο χρήστης δίνει μία εικόνα του προσώπου του και βρίσκονται παρόμοιες φωτογραφίες, και η μέθοδος στην οποία τα πρόσωπα των φωτογραφιών ομαδοποιούνται και έπειτα αντιστοιχίζονται σε κάποιο bib number (την οποία ακολουθώ και εγώ). Επίσης αναφέρει τις cloud υπηρεσίες που μπορούν να χρησιμοποιηθούν για το πρόβλημα (Google, Amazon, Microsoft). Αξίζει να σημειωθεί πως καταλήγει πως η πιο ακριβής επιλογή για την αναγνώριση των αριθμών ήταν το Amazon Rekognition.

Ακόμα υπάρχουν σελίδες , όπως η Photohawk και η myLaps/RunnerTag , οι οποίες προσφέρουν υπηρεσίες tagging των εικόνων με αναγνώριση του bib number, και αναζήτησης με βάση μία reference

εικόνα ενός δρομέα. Οι μέθοδοι που ακολουθούν ωστόσο δεν είναι σαφείς καθώς είναι εμπορικές υπηρεσίες.

1.2.2 Η δική μου προσέγγιση

Καθώς επιθυμώ οι δρομείς να μπορούν να βλέπουν και φωτογραφίες τους στις οποίες δεν είναι εμφανώς ορατό το bib number, αλλά επίσης δε θέλω να πρέπει να ανεβάζουν μία selfie ώστε να γίνεται αναζήτηση με βάση το πρόσωπο τους , αφού σε πολλούς δε θα άρεσε, καταλήγω να χρησιμοποιώ μία υβριδική τεχνική.

Χρησιμοποιώ το Amazon Rekognition, για να κάνω μία κατηγοριοποίηση των εικόνων με βάση τα bib numbers που εμφανίζονται σε αυτές, και το InsightFace, εφαρμόζωντας clustering στα πρόσωπα που εντοπίζονται , για μία δεύτερη κατηγοριοποίηση των εικόνων με βάση τα πρόσωπα που εμφανίζονται σε αυτές. Μετά συνδυάζω τα δύο αποτελέσματα για να βρω με ακρίβεια ποιες εικόνες αντιστοιχούν σε κάθε δρομέα.

Amazon Rekognition

Το **Amazon Rekognition** είναι μια υπηρεσία ανάλυσης εικόνας και βίντεο που προσφέρεται από την Amazon Web Services (AWS) και βασίζεται σε τεχνολογίες μηχανικής μάθησης και όρασης υπολογιστών (computer vision). Επιτρέπει την αυτόματη αναγνώριση αντικειμένων, προσώπων, κειμένου, σκηνών και δραστηριοτήτων σε εικόνες και βίντεο, καθώς και την ανίχνευση ακατάλληλου περιεχομένου.

Μεταξύ των βασικών λειτουργιών της περιλαμβάνονται:

- **Αναγνώριση προσώπων** (face detection, analysis, και matching),
- **Ανάλυση συναισθημάτων και χαρακτηριστικών προσώπων**,
- **Αναγνώριση κειμένου μέσα σε εικόνες** (OCR),
- **Ανίχνευση και παρακολούθηση αντικειμένων σε βίντεο**,
- **Μέθοδοι εποπτευόμενης ταυτοποίησης προσώπων** για εφαρμογές ασφάλειας και παρακολούθησης.

Το Rekognition ενδείκνυται για χρήση σε εφαρμογές όπως ταυτοποίηση ταυτότητας, έξυπνη επιτήρηση, διαχείριση μέσων, και φιλτράρισμα περιεχομένου. Συνδυάζει υψηλή ακρίβεια με εύκολη ενσωμάτωση μέσω API και αποτελεί ένα από τα πιο ολοκληρωμένα εργαλεία computer vision στο οικοσύστημα του AWS. [11]

Στην τωρινή εφαρμογή χρησιμοποιώ το Amazon Rekognition με την λειτουργία text in image, ώστε για κάθε εικόνα να βρώ τα bib numbers (τους ξεχωριστούς αριθμούς που φέρουν οι δρομείς στις μπλούζες τους) που εμφανίζονται σε αυτήν. Έτσι, αφού επεξεργαστούν όλες οι εικόνες, τελικά δημιουργώ ένα λεξικό με κλειδί το bib number κάθε δρομέα και τιμή, μία λίστα με τις φωτογραφίες στις οποίες εμφανίζεται αυτό το νούμερο.

Το κόστος για την υπηρεσία Amazon Rekognition διαμορφώνεται στα 0.0012 δολάρια ανά φωτογραφία για τις πρώτες 1 εκατομμύριο φωτογραφίες, οπότε συνολικά για μία διοργάνωση με χιλιάδες φωτογραφίες κοστίζει μόνο μερικά δολάρια.

Παράδειγμα χρήσης:

Για την παρακάτω εικόνα:

Παίρνω:

Bib: 21524, Confidence: 96.63504028320312

Bib: 5205, Confidence: 89.31879425048828

Bib: 53321, Confidence: 99.53498077392578

Bib: 50195, Confidence: 99.3216323852539

Bib: 2144, Confidence: 64.02459716796875

Ο κώδικας που έτρεξα για το παράδειγμα:

```python
import boto3


def isMarathonBib(value):

        try:
            int(value)
            return True
        except ValueError:
            return False
--

session = boto3.Session()

rekognition_client = session.client("rekognition", region_name="eu-central-1")

image_path = "/home/christos/uni_link/diplomatiki/reportFiles/testImage.jpg"


with open(image_path, "rb") as image_file:
    image_bytes = image_file.read()

response = rekognition_client.detect_text(Image={"Bytes": image_bytes})

detected_text = list(
    set(
        [
            (item["DetectedText"], item["Confidence"])
            for item in response["TextDetections"]
            if isMarathonBib(item["DetectedText"])
        ]
    )
)

for bib, confidence in detected_text:
    print(f"Bib: {bib}, Confidence: {confidence}")
```

Ωστόσο η χρήση μόνο αναγνώρισης κειμένου δεν είναι επαρκής λύση καθώς συχνά υπάρχουν εικόνες στις οποίες εμφανίζεται σαφώς ο δρομέας, αλλά δεν είναι εμφανής ο προσωπικός του αριθμός. Για να λυθεί αυτό το πρόβλημα χρειάζεται να χρησιμοποιήσουμε μοντέλα αναγνώρισης προσώπου. Το Amazon Rekognition παρέχει την δυνατότητα αναγνώρισης προσώπου, ωστόσο με περιορισμούς. Μέσω αυτού θα μπορούσαμε να βρούμε και να αποθηκεύσουμε όλα τα πρόσωπα που εμφανίζονται στο σύνολο φωτογραφιών μας και μετά να δώσουμε μία ενδεικτική εικόνα ενός προσώπου για να βρεθούν τα ταιριαστά πρόσωπα. Αυτή η λειτουργία στην περίπτωση μας δεν είναι χρήσιμη , αφού δεν έχουμε εκ των προτέρων κάποια reference εικόνα για κάθε δρομέα μέσω της οποίας να γίνει η αναζήτηση, αλλά επίσης οι δρομείς είναι πιθανό να μη θέλουν να πρέπει να δώσουν μία selfie για να δουν τις φωτογραφίες τους. Για αυτό αναζητώ λύσεις που επιτρέπουν πιο σύνθετες χρήσεις και καταλήγω στο framework InsightFace , λόγω της καλής απόδοσης του αλλά και της δημοτικότητας του.

InsightFace

Το **InsightFace** είναι ένα προηγμένο, ανοιχτού κώδικα (open-source) σύστημα αναγνώρισης προσώπων που βασίζεται σε deep learning και αναπτύχθηκε αρχικά από την κοινότητα του DeepInsight. Πρόκειται για ένα από τα πιο δημοφιλή frameworks για **face detection**, **face recognition**, **face alignment** και **face analysis**.

Βασικά χαρακτηριστικά του InsightFace:

- **Υψηλή ακρίβεια**: Το σύστημα βασίζεται σε state-of-the-art δίκτυα (όπως ArcFace, RetinaFace, MagFace κ.ά.) που έχουν επιδείξει εξαιρετική απόδοση σε benchmarks όπως το LFW και MegaFace.

- **Face Embeddings**: Μετατρέπει πρόσωπα σε αριθμητικά διανύσματα (embeddings), που επιτρέπουν αποδοτική αναζήτηση και ταυτοποίηση.

- **Ευελιξία και υποστήριξη πολλαπλών αρχιτεκτονικών**: Υποστηρίζει ResNet, MobileFaceNet, EfficientNet και άλλα, με δυνατότητα επιλογής ανάλογα με τις ανάγκες απόδοσης και ταχύτητας.

- **Face Alignment**: Διορθώνει τη θέση και γωνία του προσώπου για καλύτερη συνέπεια στην αναγνώριση.

- **Υποστήριξη GPU acceleration**: Βασίζεται σε PyTorch ή MXNet, με πλήρη υποστήριξη για hardware επιτάχυνση.

Εκμεταλλεύομαι τη δυνατότητα του InsightFace να εξάγει τα face embeddings των προσώπων που βρίσκονται σε μία εικόνα. Έτσι αφού αυτά εξαχθούν από τις εικόνες τα χωρίζω σε clusters, κάνοντας χρήση των γνωστών αλγορίθμων, ώστε κάθε cluster να αντιστοιχεί σε ένα πρόσωπο/δρομέα. [12]

Το InsightFace μας δίνεται η δυνατότητα να επιλέξουμε διαφορετικά μοντέλα για το κομμάτι του Face Detection και Face Recognition που χρειάζεται για τη περίπτωση μας. Εγώ επιλέγω το configuration buffalo_l

Name	Detection Model	Recognition Model	Alignment	Attributes	Model-Size
antelopev2	RetinaFace-10GF	ResNet100@Glint360K	2d106 & 3d68	Gender&Age	407MB
buffalo_l	RetinaFace-10GF	ResNet50@WebFace600K	2d106 & 3d68	Gender&Age	326MB
buffalo_m	RetinaFace-2.5GF	ResNet50@WebFace600K	2d106 & 3d68	Gender&Age	313MB
buffalo_s	RetinaFace-500MF	MBF@WebFace600K	2d106 & 3d68	Gender&Age	159MB
buffalo_sc	RetinaFace-500MF	MBF@WebFace600K	-	-	16MB

Σχήμα 1: InsightFace configuration

Καθώς όπως βλέπουμε από το ίδιο το github repo του InsightFace, τα μοντέλα που χρησιμοποιεί RetinaFace-10GF και ResNet50@WebFace600K έχουν την καλύτερη επίδοση:

Dataset	MR-ALL	African	Caucasian	South Asian	East Asian	LFW	CFP-FP	AgeDB-30	IJB-C(E4)	Link(onnx)
CISIA	36.794	42.550	55.825	49.618	19.611	99.450	95.214	94.900	87.220	GDrive
CISIA_pfc	37.107	38.934	53.823	48.674	19.927	99.367	95.429	94.600	84.970	GDrive
VGG2	38.578	35.259	54.304	44.081	24.095	99.550	97.410	95.080	91.220	GDrive
VGG2_pfc	40.673	36.767	60.180	49.039	24.255	99.683	98.529	95.400	92.490	GDrive
GlintAsia	62.663	49.531	64.829	57.984	61.743	99.583	93.186	95.400	91.500	GDrive
GlintAsia_pfc	63.149	50.366	65.227	57.936	61.820	99.650	93.029	95.233	91.140	GDrive
MS1MV2	77.696	74.596	84.126	82.041	51.105	99.833	98.083	98.083	96.140	GDrive
MS1MV2_pfc	77.738	74.728	84.883	82.798	52.507	99.783	98.071	98.017	96.080	GDrive
MS1M_MegaFace	78.372	74.138	82.251	77.223	60.203	99.750	97.557	97.400	95.350	GDrive
MS1M_MegaFace_pfc	78.773	73.690	82.947	78.793	57.566	99.800	97.870	97.733	95.400	GDrive
MS1MV3	82.522	77.172	87.028	86.006	60.625	99.800	98.529	98.267	96.580	GDrive
MS1MV3_pfc	81.683	78.126	87.286	85.542	58.925	99.800	98.443	98.167	96.430	GDrive
Glint360k	86.789	84.749	91.414	90.088	66.168	99.817	99.143	98.450	97.130	GDrive
Glint360k_pfc	87.077	85.272	91.616	90.541	66.813	99.817	99.143	98.450	97.020	GDrive
WebFace600K	90.566	89.355	94.177	92.358	73.852	99.800	99.200	98.100	97.120	GDrive
WebFace600K_pfc	89.951	89.301	94.016	92.381	73.007	99.817	99.143	98.117	97.010	GDrive
Average	69.247	65.908	77.121	72.819	52.014	99.706	97.374	96.962	93.925	
Average_pfc	69.519	65.898	77.497	73.213	51.853	99.715	97.457	96.965	93.818	

2.1 RetinaFace

In RetinaFace, mAP was evaluated with multi-scale testing.

`m025` : means MobileNet-0.25

Impelmentation	Easy-Set	Medium-Set	Hard-Set	Link
RetinaFace-R50	96.5	95.6	90.4	BDrive, GDrive
RetinaFace-m025(yangfly)	-	-	82.5	BDrive(nzof), GDrive
BlazeFace-FPN-SSH (paddle)	91.9	89.8	81.7%	pretrained model, inference mo

2.2 SCRFD

In SCRFD, mAP was evaluated with single scale testing, VGA resolution.

`2.5G` : means the model cost `2.5G` FLOPs while the input image is in VGA(640x480) resolution.

`_KPS` : means this model can detect five facial keypoints.

Name	Easy	Medium	Hard	FLOPs	Params(M)	Infer(ms)	Link(pth)
SCRFD_500M	90.57	88.12	68.51	500M	0.57	3.6	GDrive
SCRFD_1G	92.38	90.57	74.80	1G	0.64	4.1	GDrive
SCRFD_2.5G	93.78	92.16	77.87	2.5G	0.67	4.2	GDrive
SCRFD_10G	95.16	93.87	83.05	10G	3.86	4.9	GDrive
SCRFD_34G	96.06	94.92	85.29	34G	9.80	11.7	GDrive
SCRFD_500M_KPS	90.97	88.44	69.49	500M	0.57	3.6	GDrive
SCRFD_2.5G_KPS	93.80	92.02	77.13	2.5G	0.82	4.3	GDrive
SCRFD_10G_KPS	95.40	94.01	82.80	10G	4.23	5.0	GDrive

Παράδειγμα χρήσης του InsightFace:

Embedding 1 (first 5 values): [1.580, -0.887, 0.045, 0.609, -0.724, ...]

Embedding 2 (first 5 values): [0.583, 0.363, -0.035, 0.341, -1.066, ...]

Embedding 3 (first 5 values): [-0.993, 0.625, -1.182, 2.516, 1.694, ...]

Embedding 4 (first 5 values): [0.368, -0.898, 2.043, -0.071, 1.518, ...]

Embedding 5 (first 5 values): [0.430, 1.158, 0.764, -0.234, -0.931, ...]

Embedding 6 (first 5 values): [0.398, -0.955, 1.820, 0.153, 0.830, ...]

Embedding 7 (first 5 values): [0.340, 0.243, 0.767, 0.100, 0.646, ...]

Embedding 8 (first 5 values): [1.148, -0.967, 0.185, -1.569, -0.864, ...]

Embedding 9 (first 5 values): [-1.917, 0.385, -0.914, 0.088, 0.184, ...]

Ο κώδικας που έτρεξα για το παράδειγμα:

```python
import cv2
import insightface
from insightface.app import FaceAnalysis
import numpy as np

# Initialize face analysis
app = FaceAnalysis(name='buffalo_l')--
app.prepare(ctx_id=0)

image_path = 'testImage.jpg'--
image = cv2.imread(image_path)

faces = app.get(image)
i=1
for face in faces:
    box = face.bbox.astype(int)
    embedding = face.embedding  # 512-dimensional vector

    cv2.rectangle(image, (box[0], box[1]), (box[2], box[3]), (0, 255, 0), 2)

    print(f"Embedding {i} (first 5 values): [{', '.join(f'{x:.3f}' for x in embedding[:5])}, ...]")
    i=i+1
cv2.imwrite("output_with_faces.jpg", image)

```

Έχοντας εξάγει λοιπόν όλα τα embeddings, χρησιμοποιώ τον αλγόριθμο DBSCAN για την ομαδοποίηση τους σε clusters. Χρησιμοποιώ την έτοιμη υλοποίηση του αλγορίθμου από την open-source βιβλιοθήκη μηχανικής μάθησης Scikit-learn. Έτσι καταλήγω με ένα λεξικό με κλειδί το label του cluster και τιμή μία λίστα με τις εικόνες στις οποίες εμφανίζεται το αντίστοιχο πρόσωπο/δρομέας.

Συγχώνευση των επιμέρους αποτελεσμάτων

Πλέον έχουμε ένα λεξικό με τις εικόνες για κάθε bib number και ένα λεξικό για τις εικόνες κάθε cluster. Το πρόβλημα είναι ότι δε γνωρίζουμε ποιος cluster αντιστοιχεί σε ποιο bib number. Για να το λύσω αυτό χρησιμοποιώ μία απλή τεχνική, για κάθε cluster ελέγχω όλα τα bib numbers για να δώ με ποιο υπάρχει μεγαλύτερη επικάλυψη (κατά απόλυτο αριθμό φωτογραφιών) στην λίστα των φωτογραφιών. Έπειτα προσθέτω τις φωτογραφίες του cluster στην λίστα του bib number με το οποίο αντιστοιχήθηκε ώστε τελικά το λεξικά bib number - εικόνων, να περιέχει τις φωτογραφίες που αντιστοιχούν σε κάθε δρομέα.

Ο κώδικας για την συγχώνευση των αποτελεσμάτων και την αποθήκευση τους στο DynamoDB:

```python
import json
import boto3

with open('bibImages.json', 'r') as f:
    bibImages = json.load(f)

with open('clusterImages.json', 'r') as f:
    clusterImages = json.load(f)


bibImages = {int(bib): images for bib, images in bibImages.items()}
clusterImages = {int(cluster): images for cluster, images in clusterImages.items()}

for bib, bib_imgs in bibImages.items():
    best_cluster = None
    max_overlap = 0
    best_cluster_imgs = []
    for cluster, cluster_imgs in clusterImages.items():
        overlap = len(set(cluster_imgs) & set(bib_imgs))  # Count common images
        if overlap > max_overlap and cluster!=-1:
            max_overlap = overlap
            best_cluster = cluster
            best_cluster_imgs = cluster_imgs

    if best_cluster is not None:
        bibImages[bib] = list(set(bibImages[bib]).union(best_cluster_imgs))
        print(f"Cluster {best_cluster} matched with bib {bib} (overlap: {max_overlap})")
    else:
        print(f"Cluster {bib} had no matching cluster")



session = boto3.Session()
dynamodb = session.resource('dynamodb', region_name='eu-central-1')

table = dynamodb.Table('bib-images.predictions2024')
for bib , images in bibImages.items():
    table.put_item(
        Item={
            'bib': str(bib),
            'images': images
        }
    )
```

Κεφάλαιο 2 Παρουσίαση Θέματος

Κύριος στόχος είναι η ανάπτυξη μιας εφαρμογής στην οποία ο χρήστης θα μπορεί να βλέπει τις εικόνες που λήφθηκαν κατά τη διάρκεια του αγώνα, στις οποίες είναι μέσα, και να βλέπει τα δεδομένα σχετικά με την επίδοση του. Ακόμα με σκοπό να βελτιωθεί η εμπειρία χρήσης της εφαρμογής είναι χρήσιμο να προστεθούν επιπλέον λειτουργίες στην εφαρμογή οι οποίες θα προσελκύουν το ενδιαφέρον των χρηστών. Για αυτό προστίθενται δυνατότητες όπως , η ανάγνωση ενός AI-Generated comment σχετικά με την επίδοση κάθε δρομέα, ένα animation που δείχνει “σε επανάληψη” την εξέλιξη του αγώνα καθώς και μία κεντρική σελίδα για όλους τους δρομείς με γενικό leaderboard καθώς και στατιστικά για τον αγώνα.

2.1 Τα components της εφαρμογής

Κεντρική σελίδα:

Leaderboard
Το στοιχείο αυτό χρειάζεται σαν είσοδο τα βασικά δεδομένα όλων των δρομέων (αγώνας που έτρεξαν, ηλικία, φύλο, όνομα, χρόνο τερματισμού) και τους παρουσιάζει σε μία κατάταξη από την καλύτερη στην χειρότερη επίδοση , ενώ υπάρχει και η δυνατότητα εφαρμογής δυναμικών φίλτρων για την ηλικία, το φύλο και τον αγώνα συμμετοχής. Ακόμα ο χρήστης της εφαρμογής μπορεί να επιλέξει κάποιον δρομέα από την κατάταξη και να πάει στην προσωπική του σελίδα.

Statistics
Ένα μέρος της εφαρμογής στο οποίο φαίνονται διαγράμματα σχετικά με το σύνολο των δρομέων. Διαγράμματα σχετικά με την ηλικιακή κατανομή των συμμετεχόντων αλλά και τα πιο δημοφιλή clubs. Για την εξαγωγή των διαγραμμάτων γίνεται προεπεξεργασία του συνόλου των δεδομένων.

Runner Search
Στοιχείο που έχει πρόσβαση στο σύνολο των δρομέων αναφορικά με το ονοματεπώνυμο τους και το bib number τους, και επιτρέπει την fuzzy αναζήτηση χρησιμοποιώντας οποιοδήποτε από αυτά , καθώς και εμφανίζει προτάσεις με βάση το τί έχει πληκτρολογήσει ήδη ο χρήστης. Μόλις ο χρήστης εισάγει στο πεδίο το pattern αναζήτησης μπορεί να επιλέξει έναν από τους προτεινόμενους δρομείς και να πάει στην προσωπική του σελίδα.

Σελίδα του δρομέα:

Image Gallery
Το component το οποίο αναλαμβάνει την λήψη των εικόνων για τον συγκεκριμένο δρομέα από το μέσο αποθήκευσης και την παρουσίαση τους στον χρήστη.

AI-Generated Comment
Το component αυτό παίρνει ως είσοδο τα δεδομένα σχετικά με την απόδοση του χρήστη αλλά και τις πληροφορίες σχετικά με αυτόν που έχουν συλλεχθεί από τους διοργανωτές, και με βάση αυτά επικοινωνεί με το LLM μοντέλο και παρουσιάζει στον χρήστη το παραγόμενο κείμενο.

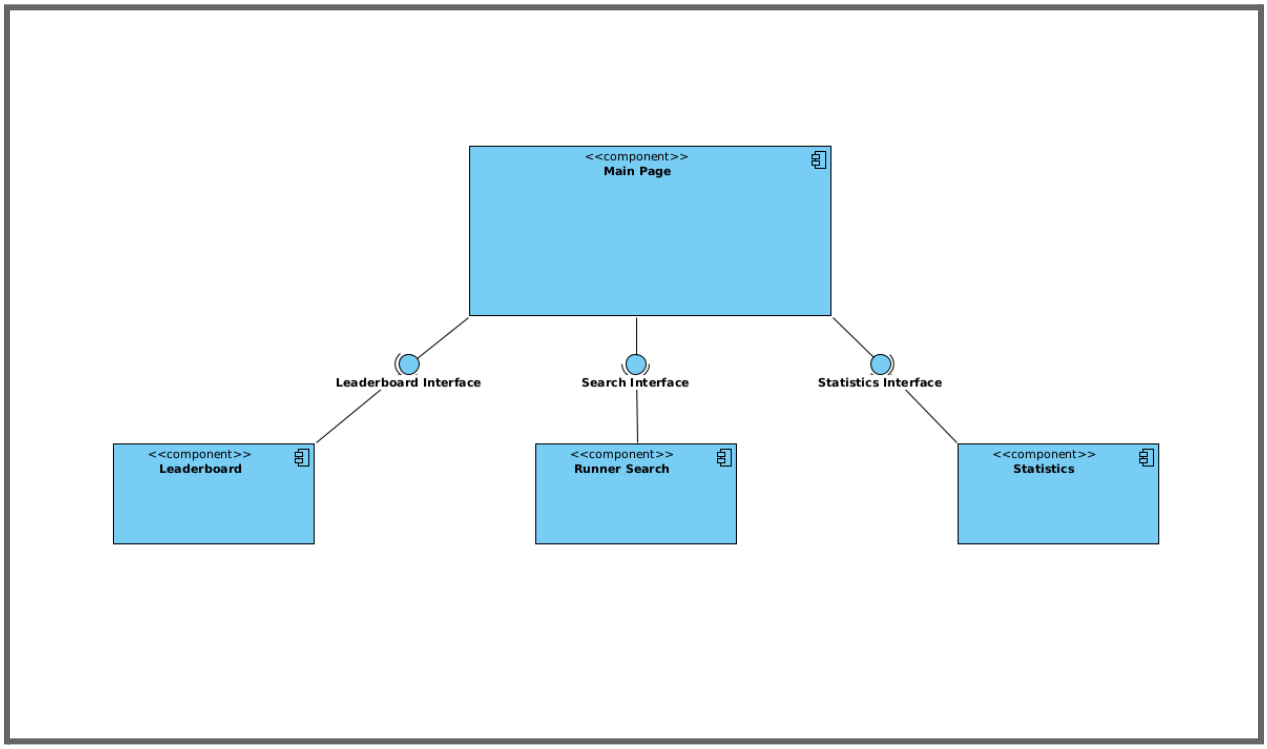
Runner Info
Αναλαμβάνει την λήψη και παρουσίαση των καταγεγραμμένων δεδομένων για τον δρομέα σχετικά με την απόδοση του , την ηλικία του, την χώρα του κλπ. Αποτελεί το "profile" του δρομέα.

Race Replay
Το συγκεκριμένο στοιχείο τοποθετεί όλους τους δρομείς του αγώνα σε έναν χάρτη στο σημείο διεξαγωγής του. Χρησιμοποιώντας τα χρονικά δεδομένα από τις μετρήσεις που έγιναν κατα τη διάρκεια του αγώνα σε διάφορα σημεία , αλλά και τις γεωγραφικές συντεταγμένες αυτών των σημείων, όπως ορίστηκαν από τους διοργανωτές, δημιουργεί κατά προσέγγιση μία πορεία την οποία ακολούθησε κάθε δρομέας. Με το σύνολο αυτών των πορειών δημιουργεί ένα animation, με τους δρομεί να κινούνται πάνω στην διαδρομή του αγώνα, ο καθένας με τον δικό του ρυθμό, επιτρέποντας στον χρήστη να φανταστεί πως θα ταν να έβλεπε τον αγώνα από ψηλά. Οι δρομείς αναπαρίστανται σαν τελείες, με την τελεία του χρήστη της εφαρμογής να ξεχωρίζει με διαφορετικό χρώμα.

2.2 Component Diagrams

Παρακάτω φαίνονται διαγράμματα που αναπαριστούν την σχέση μεταξύ των διαφόρων UI components της εφαρμογής.

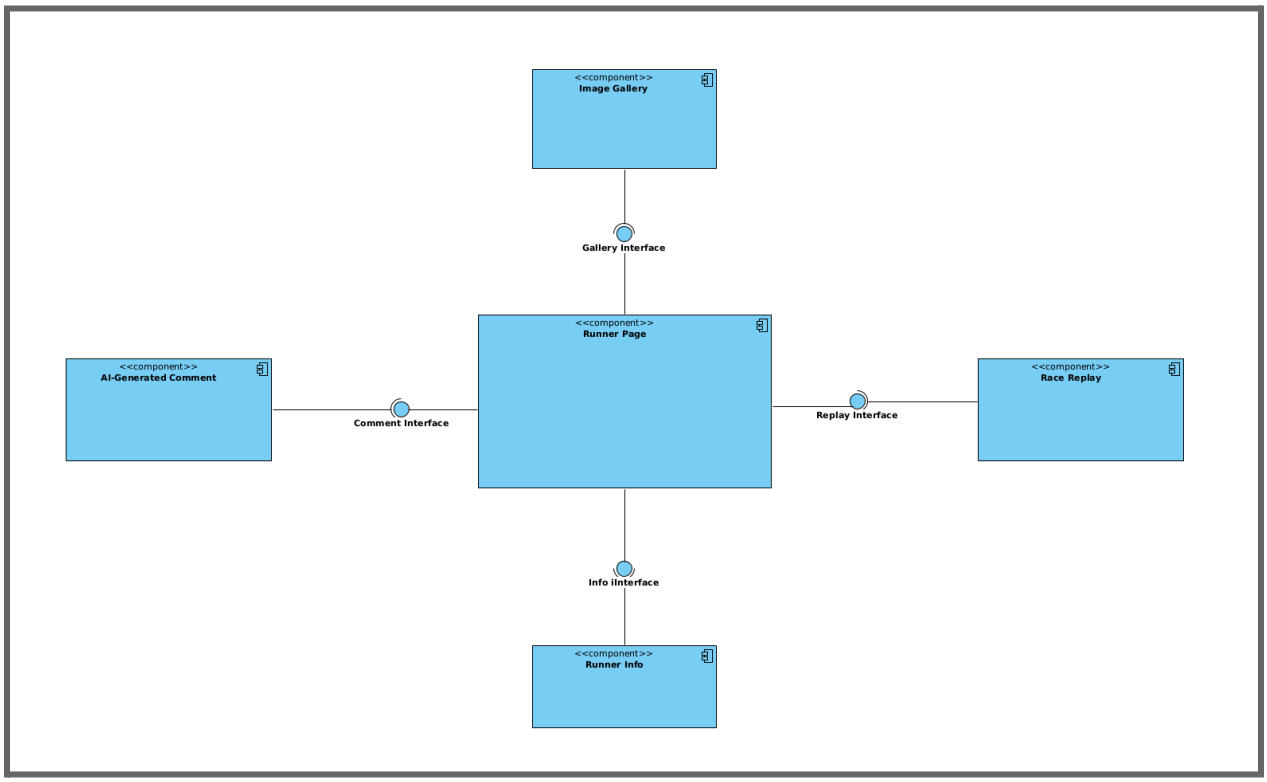
Για την κεντρική σελίδα:



Σχήμα 2: Main Page UI component diagram

Η σελίδα χρησιμοποιεί τα components παρέχοντας τους την πληροφορία που χρειάζονται και οργανώνει την παρουσίαση τους.

Για την σελίδα του δρομέα:



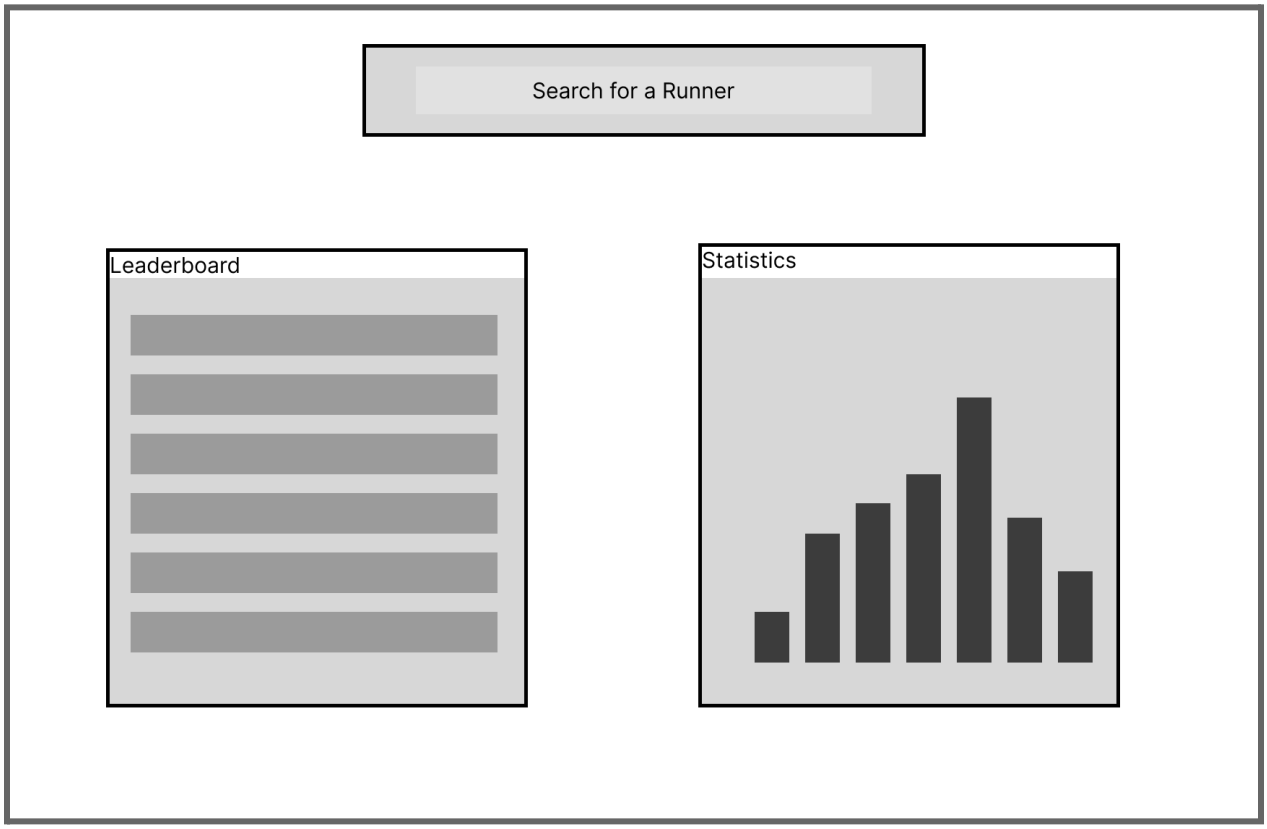
Σχήμα 3: Runner Page Component Diagram

Η σελίδα χρησιμοποιεί τα components παρέχοντας τους την πληροφορία που χρειάζονται και οργανώνει την παρουσίαση τους.

2.3 Frontend Mockups

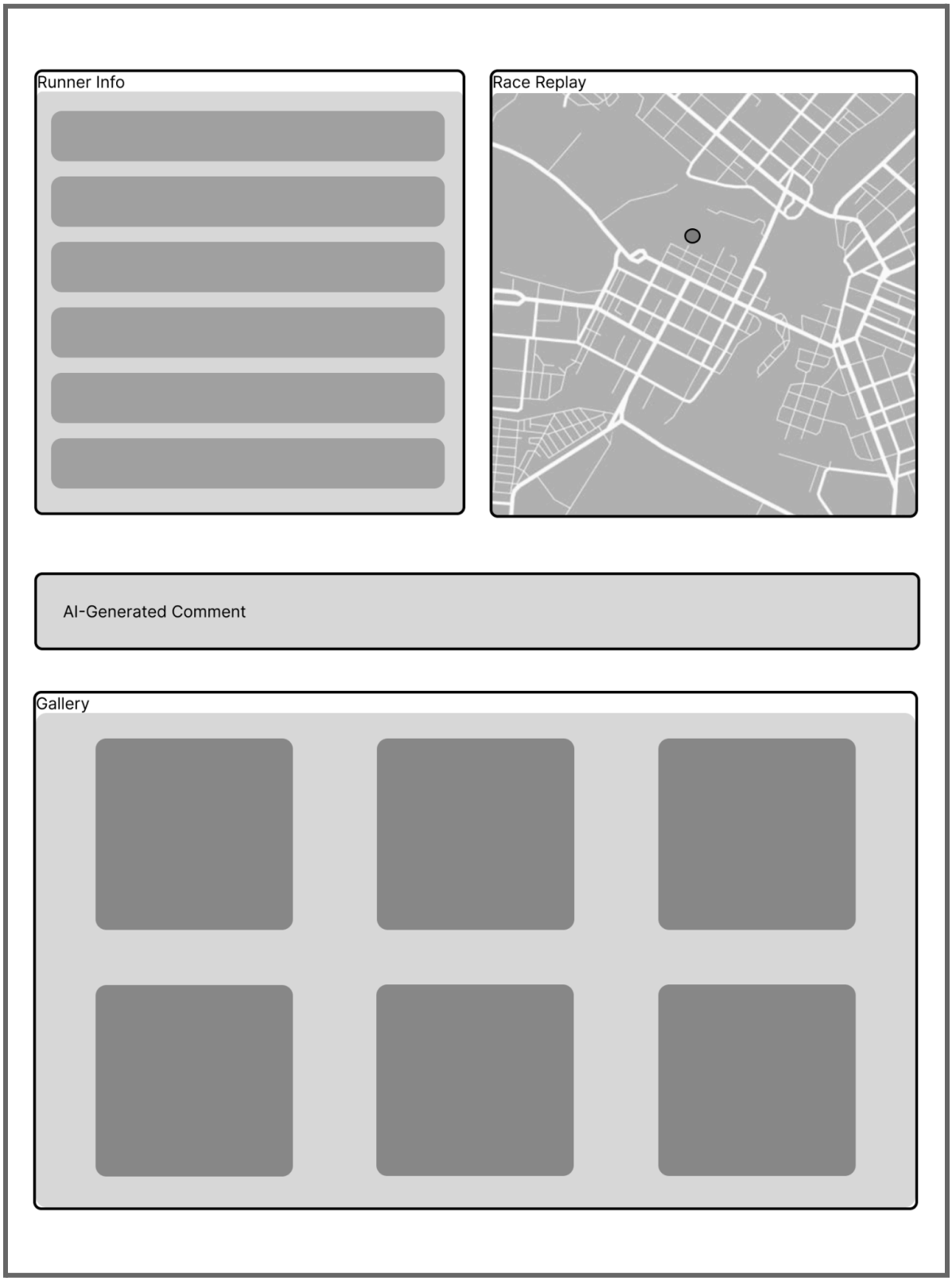
Προτού ξεκινήσει η ανάπτυξη της εφαρμογής , σχεδιάζω τα frontend mockups για τις δύο σελίδες για να έχω μία ιδέα για το πως θα είναι το ui της εφαρμογής.

Κεντρική σελίδα:



Σχήμα 4: Frontend Mockup για Main Page

Σελίδα δρομέα:



Σχήμα 5: Frontend Mockup για Runner Page

Κεφάλαιο 3 Τεχνολογικές λύσεις

3.1 Γλώσσες Προγραμματισμού και βιβλιοθήκες

Typescript

Η **TypeScript** είναι μια υπερσύνολο της JavaScript που προσθέτει στατική τυποποίηση (static typing) στη γλώσσα, επιτρέποντας τον εντοπισμό σφαλμάτων κατά τη διάρκεια της ανάπτυξης αντί για την εκτέλεση. Αναπτύχθηκε από τη Microsoft και χρησιμοποιείται ευρέως σε σύγχρονες web εφαρμογές, καθώς βελτιώνει την αξιοπιστία, την αναγνωσιμότητα και τη συντηρησιμότητα του κώδικα. Με την TypeScript, οι προγραμματιστές μπορούν να ορίζουν ρητά τύπους μεταβλητών, παραμέτρων και επιστρεφόμενων τιμών, ενώ παράλληλα επωφελούνται από δυνατότητες όπως αυτόματη συμπλήρωση κώδικα (autocompletion), έλεγχο σφαλμάτων (type checking) και ισχυρή υποστήριξη από τα περισσότερα IDEs. Ο κώδικας TypeScript μεταγλωττίζεται (transpiles) σε JavaScript, κάνοντάς τον πλήρως συμβατό με όλα τα σύγχρονα προγράμματα περιήγησης και πλατφόρμες. [13]

ReactJs

Η **React** είναι μια δημοφιλής βιβλιοθήκη JavaScript (λειτουργεί και στη Typescript) που αναπτύχθηκε από το Facebook και χρησιμοποιείται για την κατασκευή διαδραστικών και δυναμικών διεπαφών χρήστη (UI) σε web εφαρμογές. Βασίζεται στην έννοια των επαναχρησιμοποιήσιμων components, τα οποία επιτρέπουν την εύκολη οργάνωση και διαχείριση της διεπαφής, βελτιώνοντας τόσο την απόδοση όσο και την ευκολία συντήρησης του κώδικα. Χάρη στον εικονικό DOM (Virtual DOM), η React προσφέρει ταχύτατες ενημερώσεις στο περιβάλλον χρήστη, χωρίς περιττές ανανεώσεις της σελίδας. Επιπλέον, η React είναι ευέλικτη και μπορεί να ενσωματωθεί εύκολα με άλλες βιβλιοθήκες ή frameworks, ενώ υποστηρίζεται ευρέως από την κοινότητα και παρέχει πλούσια εργαλεία για ανάπτυξη, testing και debugging. [14]

NextJs

Το **Next.js** είναι ένα ισχυρό framework βασισμένο στη React, το οποίο διευκολύνει την ανάπτυξη σύγχρονων web εφαρμογών με βελτιωμένη απόδοση και καλύτερη εμπειρία χρήστη. Αναπτύχθηκε από την Vercel και προσφέρει προηγμένες δυνατότητες όπως server-side rendering (SSR), static site generation (SSG) και client-side rendering, δίνοντας τη δυνατότητα στον προγραμματιστή να επιλέξει την καταλληλότερη στρατηγική ανάλογα με τις ανάγκες της εφαρμογής. Το Next.js ενσωματώνει επίσης αυτόματο routing, υποστήριξη για API routes, βελτιστοποίηση εικόνων και εύκολη ενσωμάτωση με TypeScript. Χάρη σε αυτά τα χαρακτηριστικά, αποτελεί μια ολοκληρωμένη λύση για τη δημιουργία ταχύτατων, SEO-friendly και επεκτάσιμων εφαρμογών, αξιοποιώντας πλήρως τις δυνατότητες της React με λιγότερη πολυπλοκότητα στην αρχιτεκτονική. [15]

AntD

Το **Ant Design (AntD)** είναι μια δημοφιλής βιβλιοθήκη component UI για εφαρμογές React, σχεδιασμένη με στόχο τη δημιουργία κομψών, συνεπών και φιλικών προς τον χρήστη διεπαφών, ιδίως για enterprise-level εφαρμογές. Παρέχει ένα πλούσιο σύνολο έτοιμων στοιχείων (components) όπως φόρμες, πίνακες, κουμπιά, ειδοποιήσεις, μενού πλοήγησης κ.ά., τα οποία είναι εύκολα παραμετροποιήσιμα και έτοιμα προς χρήση. Το AntD βασίζεται στις αρχές του Design System και προσφέρει πλήρη υποστήριξη για προσβασιμότητα (accessibility), localization και responsive design. Επιπλέον, διαθέτει εκτενή τεκμηρίωση και υποστήριξη από την κοινότητα, καθιστώντας το μια σταθερή επιλογή για γρήγορη και επαγγελματική ανάπτυξη React εφαρμογών με συνεπή εμφάνιση και λειτουργικότητα. [16]

Tailwind Css

Το Tailwind CSS είναι ένα utility-first framework για τη δημιουργία γρήγορων και προσαρμόσιμων user interfaces απευθείας μέσα στο HTML ή JSX κώδικα. Αντί να βασίζεται σε προκαθορισμένα components ή έτοιμα στυλ, το Tailwind παρέχει ένα ευρύ σύνολο από χαμηλού επιπέδου κλάσεις (utilities) για στυλ όπως χρώματα, αποστάσεις, περιθώρια, γραμματοσειρές, ευθυγραμμίσεις κ.ά., επιτρέποντας στον προγραμματιστή να σχεδιάζει διεπαφές χωρίς να γράφει καθόλου custom CSS. Υποστηρίζει επίσης responsive design, dark mode, animation και μπορεί να παραμετροποιηθεί πλήρως μέσω αρχείων ρυθμίσεων. Χάρη στην απλότητά του και την ταχύτητα που προσφέρει στην ανάπτυξη, το Tailwind CSS έχει γίνει εξαιρετικά δημοφιλές σε σύγχρονες web εφαρμογές, ιδίως σε συνδυασμό με React και Next.js. [17]

Python

Η **Python** είναι μια υψηλού επιπέδου, δυναμικά τυποποιημένη γλώσσα προγραμματισμού που φημίζεται για την απλότητα και αναγνωσιμότητα του συντακτικού της. Υποστηρίζει πολλαπλά παραδείγματα προγραμματισμού, όπως αντικειμενοστραφή, διαδικαστικό και λειτουργικό προγραμματισμό, γεγονός που την καθιστά ευέλικτη και κατάλληλη για μια μεγάλη ποικιλία εφαρμογών. Η Python χρησιμοποιείται ευρέως σε τομείς όπως η ανάλυση δεδομένων, η τεχνητή νοημοσύνη, η μηχανική μάθηση, η ανάπτυξη web, αλλά και στην αυτοματοποίηση συστημάτων. Διαθέτει τεράστια κοινότητα και πλούσιο οικοσύστημα βιβλιοθηκών και εργαλείων, γεγονός που επιταχύνει την ανάπτυξη και διευκολύνει τη συντήρηση του λογισμικού. Χάρη στην απλότητά της, είναι επίσης ιδιαίτερα δημοφιλής ως γλώσσα εκμάθησης για νέους προγραμματιστές. Στα πλαίσια της εφαρμογής μου χρησιμοποιείται για στα lambda functions. [18]

3.2 Υπηρεσίες cloud

Cloud

Το **cloud** (υπολογιστικό νέφος) είναι η τεχνολογία που επιτρέπει την αποθήκευση, επεξεργασία και πρόσβαση σε δεδομένα και εφαρμογές μέσω του διαδικτύου, χωρίς την ανάγκη τοπικής εγκατάστασης ή χρήσης φυσικών υπολογιστικών πόρων από τον χρήστη. Οι υπηρεσίες cloud προσφέρονται από εξειδικευμένους παρόχους και επιτρέπουν στους χρήστες να χρησιμοποιούν ισχυρά υπολογιστικά συστήματα, να αποθηκεύουν αρχεία και να τρέχουν εφαρμογές από οπουδήποτε και οποιαδήποτε στιγμή, αρκεί να υπάρχει σύνδεση στο διαδίκτυο. Το cloud χρησιμοποιείται ευρέως τόσο από ιδιώτες όσο και από επιχειρήσεις λόγω της ευελιξίας, της ασφάλειας και του χαμηλότερου κόστους συντήρησης που προσφέρει.

Για τις ανάγκες της εφαρμογής χρησιμοποιώ τις τεχνολογίες του cloud της amazon (Amazon Web Services).

Amazon Web Services

Το **Amazon Web Services (AWS)** είναι η πλατφόρμα cloud της Amazon και αποτελεί μία από τις πιο δημοφιλείς και ευρέως χρησιμοποιούμενες υπηρεσίες υπολογιστικού νέφους παγκοσμίως. Μέσω του AWS, η Amazon προσφέρει μια μεγάλη γκάμα υπηρεσιών όπως αποθήκευση δεδομένων, υπολογιστική ισχύ (π.χ. μέσω των servers EC2), βάσεις δεδομένων, τεχνητή νοημοσύνη, μηχανική μάθηση και πολλά άλλα, όλα διαθέσιμα μέσω του διαδικτύου. Οι χρήστες – από startups μέχρι μεγάλες επιχειρήσεις και κρατικούς οργανισμούς – μπορούν να "νοικιάζουν" αυτούς τους πόρους ανάλογα με τις ανάγκες τους, αποφεύγοντας το κόστος αγοράς και συντήρησης φυσικών υπολογιστικών υποδομών. Το AWS είναι γνωστό για την αξιοπιστία, την κλιμακωσιμότητα, την ασφάλεια και την παγκόσμια κάλυψή του, προσφέροντας data centers σε πολλές περιοχές του κόσμου.

Υπηρεσίες της Amazon που χρησιμοποιήθηκαν

Amazon S3

Το **Amazon S3 (Simple Storage Service)** είναι μια υπηρεσία αποθήκευσης αντικειμένων (object storage) που παρέχεται από το Amazon Web Services (AWS) και έχει σχεδιαστεί για να προσφέρει υψηλή διαθεσιμότητα, ασφάλεια και επεκτασιμότητα. Επιτρέπει στους χρήστες να αποθηκεύουν και να ανακτούν οποιαδήποτε ποσότητα δεδομένων, οποιαδήποτε στιγμή και από οπουδήποτε στο διαδίκτυο. Τα δεδομένα αποθηκεύονται σε μορφή "αντικειμένων" μέσα σε "buckets" και μπορούν να περιλαμβάνουν αρχεία κάθε τύπου, όπως εικόνες, βίντεο, έγγραφα ή backup αρχείων. Το Amazon S3 χρησιμοποιείται ευρέως για backup, αποθήκευση αρχείων εφαρμογών, φιλοξενία ιστοσελίδων και διανομή περιεχομένου. Χαρακτηρίζεται από υψηλή ασφάλεια, επιλογές για έλεγχο πρόσβασης, καθώς και υποστήριξη versioning και replication, καθιστώντας το μια αξιόπιστη λύση για επιχειρήσεις και προγραμματιστές. [19]

Amazon DynamoDB

Το **Amazon DynamoDB** είναι μια πλήρως διαχειριζόμενη, μη σχεσιακή βάση δεδομένων (NoSQL) που προσφέρεται από το Amazon Web Services (AWS) και έχει σχεδιαστεί για εφαρμογές που απαιτούν υψηλή απόδοση, ταχύτητα και ευελιξία στην κλιμάκωση. Υποστηρίζει δομές τύπου key-value και έγγραφα, και επιτρέπει την αποθήκευση και την ανάκτηση δεδομένων με πολύ χαμηλή καθυστέρηση, ακόμη και σε πολύ μεγάλες κλίμακες. Το DynamoDB προσφέρει αυτόματη κλιμάκωση χωρητικότητας, υψηλή διαθεσιμότητα, ασφάλεια και ενσωμάτωση με άλλες υπηρεσίες του AWS. Χρησιμοποιείται ευρέως σε εφαρμογές real-time, όπως παιχνίδια, συστήματα ηλεκτρονικού εμπορίου, mobile apps και IoT, όπου η ταχύτητα και η αξιοπιστία στη διαχείριση των δεδομένων είναι κρίσιμη. [20]

Amazon Lambda

Η **Amazon Lambda** είναι μια υπηρεσία υπολογιστικού νέφους που προσφέρεται από το Amazon Web Services (AWS) και επιτρέπει την εκτέλεση κώδικα χωρίς τη διαχείριση ή την παροχή διακομιστών – γι' αυτό και χαρακτηρίζεται ως **serverless**. Με την Amazon Lambda, οι προγραμματιστές μπορούν να ανεβάσουν τον κώδικά τους και να τον εκτελέσουν αυτόματα ως απόκριση σε διάφορα γεγονότα (events), όπως αλλαγές σε ένα Amazon S3 bucket, νέες εγγραφές σε μια βάση δεδομένων DynamoDB ή αιτήματα μέσω του API Gateway.

Η υπηρεσία χειρίζεται αυτόματα την κλιμάκωση, την παρακολούθηση, την ασφάλεια και τη συντήρηση της υποδομής, επιτρέποντας στους χρήστες να επικεντρωθούν μόνο στη λογική του προγράμματος. Οι χρεώσεις βασίζονται μόνο στον χρόνο που τρέχει ο κώδικας και στον αριθμό των αιτήσεων, καθιστώντας την Amazon Lambda ιδιαίτερα οικονομική για πολλές εφαρμογές, όπως μικροϋπηρεσίες (microservices), αυτοματοποιήσεις, real-time επεξεργασία δεδομένων, και backend για εφαρμογές web ή κινητών. [21]

Amazon API Gateway

Το **Amazon API Gateway** είναι μια πλήρως διαχειριζόμενη υπηρεσία της Amazon Web Services (AWS**)** που επιτρέπει τη δημιουργία, έκδοση, διαχείριση και ασφάλεια API (Application Programming

Interfaces) σε μεγάλη κλίμακα. Μέσω του API Gateway, οι προγραμματιστές μπορούν να δημιουργούν RESTful ή WebSocket APIs που λειτουργούν ως «γέφυρα» ανάμεσα σε πελάτες (όπως εφαρμογές web ή κινητών) και backend υπηρεσίες, όπως AWS Lambda functions, βάσεις δεδομένων ή άλλες εφαρμογές στο cloud.

Η υπηρεσία προσφέρει ενσωματωμένες λειτουργίες όπως αυθεντικοποίηση και εξουσιοδότηση (authentication & authorization) μέσω AWS IAM, Cognito ή custom tokens, ρυθμίσεις throttling, rate limiting, και caching για βελτίωση της απόδοσης. Επίσης, παρέχει παρακολούθηση (monitoring) και logging μέσω του Amazon CloudWatch.

Το API Gateway είναι ιδιαίτερα χρήσιμο για την υλοποίηση serverless αρχιτεκτονικών, καθώς συνεργάζεται άψογα με το AWS Lambda, επιτρέποντας την κατασκευή ισχυρών, κλιμακούμενων και ασφαλών backend συστημάτων χωρίς την ανάγκη διαχείρισης servers. [22]

Amazon SAM

Το **AWS SAM (Serverless Application Model)** είναι ένα ανοικτού κώδικα framework που διευκολύνει τον σχεδιασμό, την ανάπτυξη και τη διαχείριση serverless εφαρμογών στο περιβάλλον του Amazon Web Services. Μέσω ενός απλού αρχείου YAML, το SAM επιτρέπει την περιγραφή των λειτουργιών και της υποδομής της εφαρμογής, όπως οι AWS Lambda functions, το API Gateway και οι βάσεις δεδομένων DynamoDB. Επιπλέον, προσφέρει δυνατότητες τοπικού testing και debugging, καθώς και αυτοματοποιημένο deployment μέσω του AWS CloudFormation. Με αυτόν τον τρόπο, το AWS SAM απλοποιεί και επιταχύνει τη δημιουργία serverless εφαρμογών, παρέχοντας ταυτόχρονα πλήρη έλεγχο και ευκολία στη διαχείριση της υποδομής. [23]

Amazon Amplify

Το **AWS Amplify** είναι μια πλατφόρμα ανάπτυξης εφαρμογών της Amazon Web Services που απλοποιεί τη δημιουργία, ανάπτυξη και διαχείριση σύγχρονων web και mobile εφαρμογών με ενσωματωμένες δυνατότητες cloud. Μέσω του Amplify, οι προγραμματιστές μπορούν εύκολα να προσθέσουν λειτουργίες όπως πιστοποίηση χρηστών, αποθήκευση δεδομένων, APIs, hosting και ανάλυση δεδομένων, χωρίς να χρειάζεται να διαχειρίζονται την υποδομή του cloud. Η πλατφόρμα παρέχει εργαλεία όπως το CLI, βιβλιοθήκες UI και κονσόλα διαχείρισης, που υποστηρίζουν δημοφιλή frameworks και γλώσσες, καθιστώντας το ιδανικό για ταχύτατη ανάπτυξη εφαρμογών με πλήρη αξιοποίηση των υπηρεσιών AWS. [24]

Amazon Bedrock

Η υπηρεσία Amazon Bedrock είναι μια πλήρως διαχειριζόμενη υπηρεσία της Amazon Web Services (AWS) που επιτρέπει σε επιχειρήσεις και προγραμματιστές να δημιουργούν και να ενσωματώνουν εφαρμογές τεχνητής νοημοσύνης με χρήση γενετικών μοντέλων (foundation models) από κορυφαίους

παρόχους, όπως η AI21 Labs, η Anthropic, η Cohere, η Meta, η Stability AI και η ίδια η Amazon (με τα μοντέλα Titan). Μέσω του Bedrock, οι χρήστες μπορούν να αξιοποιούν ισχυρά εργαλεία όπως η φυσική γλώσσα, η δημιουργία εικόνων, η ανάλυση δεδομένων και η παραγωγή περιεχομένου χωρίς να χρειάζονται εξειδικευμένες γνώσεις στην υποδομή ή στην εκπαίδευση μοντέλων μηχανικής μάθησης. Η υπηρεσία προσφέρει ευκολία, κλιμάκωση και ασφάλεια, ενσωματώνοντας τα οφέλη του cloud περιβάλλοντος της AWS. [25]

Amazon Rekognition

Όπως παρουσιάστηκε στο κεφάλαιο 1.2.

3.3 Υλοποίηση των components της εφαρμογής

Γενικά για την εφαρμογή

Αρχικά προτού ξεκινήσει η ανάπτυξη της εφαρμογής τοποθέτησα όλες τις unlabeled φωτογραφίες από το event σε ένα S3 bucket, ώστε να είναι προσβάσιμες από εκεί.

Ακόμα δημιούργησα ένα dynamodb table “RunnerTimes” για τα δεδομένα σχετικά με τους δρομείς, για πληροφορίες όπως το ονοματεπώνυμο, την ηλικία, τον συνολικό χρόνο τερματισμού, την κατάταξη και άλλα.

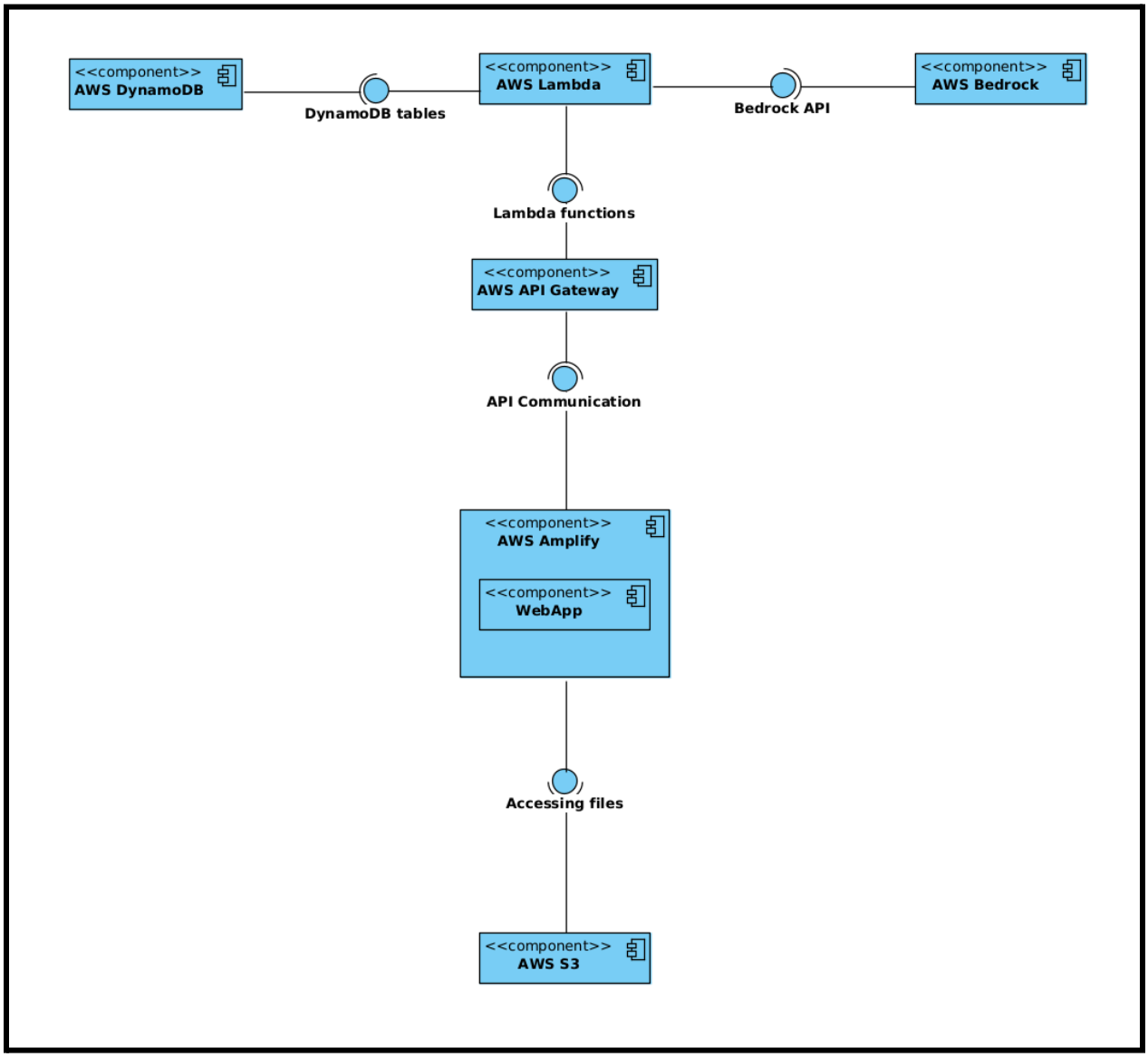
Στη συνέχεια εκτέλεσα τα αρχεία κώδικα που υλοποιούν την διαδικασία που περιγράφηκε στο κεφάλαιο 1.2 και αποθήκευσα το αποτέλεσμα σε ένα dynamodb table “bib-images2024”, ώστε να είναι απευθείας προσβάσιμα από εκεί.

Ακόμα επεξεργάστηκα τα δεδομένα των δρομέων ώστε να βρω τον αριθμό δρομέων καθώς και τον μέσο χρόνο τερματισμού για κάθε ηλικιακό γκρουπ αλλά και για κάθε ομάδα αθλητών (πχ Palaio Faliro Runners). Με αυτά τα δεδομένα δημιούργησα ένα json το οποίο αποθήκευσα στο s3, για εύκολη και γρήγορη πρόσβαση.

Δημιουργώ ένα HTTP API μέσω του Amazon API Gateway, το οποίο λειτουργεί ως backend της εφαρμογής. Μέσω αυτού του API εκτίθενται τα AWS Lambda functions, ώστε να είναι προσβάσιμα από τα frontend μέρη της εφαρμογής. Με αυτόν τον τρόπο επιτυγχάνεται αποδοτική και κλιμακούμενη επικοινωνία μεταξύ frontend και backend, χωρίς την ανάγκη διαχείρισης υποδομής.

Τέλος, χρησιμοποιώ το Amazon Amplify για τη φιλοξενία του frontend της εφαρμογής, συνδέοντας το GitHub repository του project μου. Με αυτόν τον τρόπο, το Amplify εξασφαλίζει ότι σε κάθε push στον κώδικα πραγματοποιείται αυτόματα ένα νέο deployment, με αποτέλεσμα οι αλλαγές να είναι άμεσα ορατές στη δημόσια σελίδα της εφαρμογής. Έτσι, επιτυγχάνεται μια πλήρως αυτοματοποιημένη ροή CI/CD (Continuous Integration / Continuous Deployment).

Το component diagram που δείχνει την αρχιτεκτονική της εφαρμογής:



Σχήμα 6: Backend component diagram της εφαρμογής

Home Page

Στη σελίδα αυτή τοποθετούνται τα components Leaderboard, Statistics και Runner Search, η λογική για τα οποία υλοποιείται εξολοκλήρου σε αυτά και η σελίδα απλά φροντίζει για την παρουσίαση.

Leaderboard

Για το leaderboard αρχικά υλοποιώ ένα lambda function (σε python), το οποίο διαβάζει το table “RunnerTimes” . Δέχεται παραμέτρους σχετικά με το ποιο από τα 3 events (ημιμαραθώνιος, 10 και 5 χλμ) , ποιο φύλο, και ποιο διάστημα ηλικίας του δρομέα μας ενδιαφέρει , και επιστρέφει για τους δέκα πρώτους δρομείς που ικανοποιούν αυτά τα φίλτρα, τα columns για την ηλικία, το ονοματεπώνυμο, το bib number και την επίδοση τους.

Αυτό το lambda function γίνεται deploy με το sam , και είναι ένα διαθέσιμο endpoint στο API της εφαρμογής.

Το αρχείο template.yaml της sam εφαρμογής για το deployment του lambda function μέσω του api, αντίστοιχα προσθέτονται τα υπόλοιπα functions και τελικά γίνονται deploy με το sam cli:

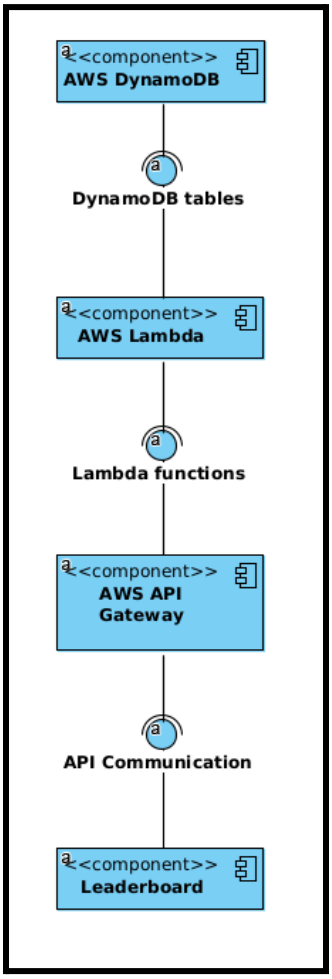
```yaml
AWSTemplateFormatVersion: "2010-09-09"
Transform: "AWS::Serverless-2016-10-31"

Resources:
  MyHttpApi:
    Type: AWS::Serverless::HttpApi
    Properties:
      StageName: Prod
      CorsConfiguration:
        AllowOrigins:
          - "*"
        AllowMethods:
          - OPTIONS
          - POST
        AllowHeaders:
          - Content-Type
  GetTopRunners:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: getTopRunners/
      Runtime: python3.12
      Handler: index.lambda_handler
      Policies:
        - AmazonDynamoDBReadOnlyAccess
      Events:
        RunnerAPI:
          Type: HttpApi
          Properties:
            Path: /getTopRunners
            Method: GET
            ApiId: !Ref MyHttpApi
```

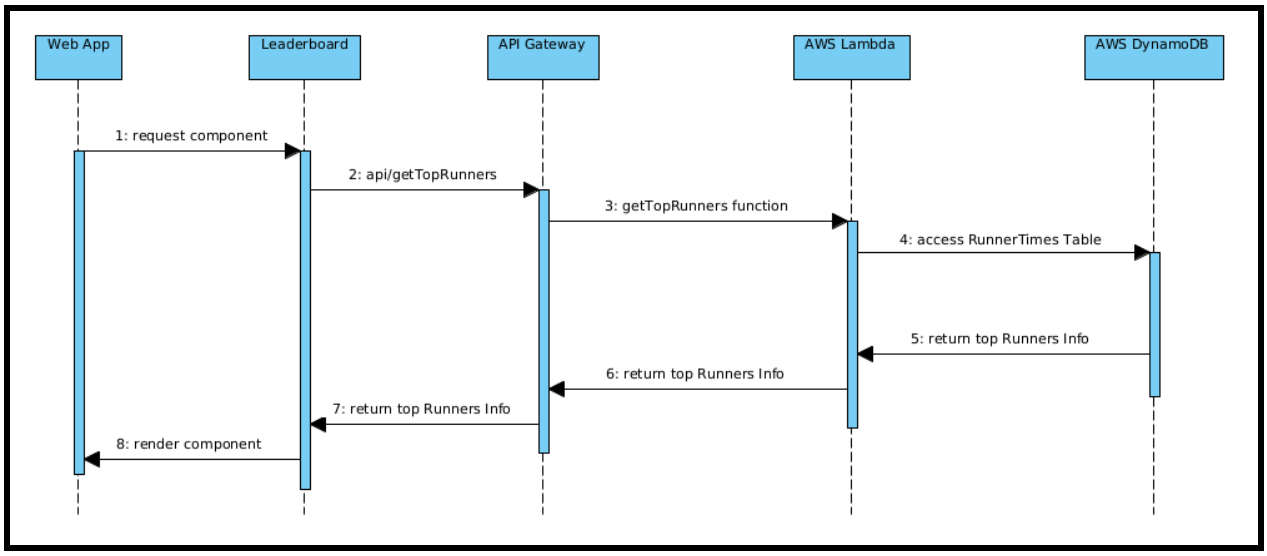
Το react component την πρώτη φορά που γίνεται render, ή όταν ο χρήστης αλλάζει κάποιο από τα φίλτρα (useEffect() hook), στέλνει request σε αυτό το endpoint και ανανεώνει το ui με βάση το response. Ακόμα όταν ο χρήστης επιλέξει έναν δρομέα από αυτή τη κατάταξη πάει στη σελίδα του (useRouter() hook).

Για το ui χρησιμοποιούνται components όπως το Card, Table, Slider από το antD.

Παρακάτω φαίνονται τα component και sequence diagrams για το component του Leaderboard:

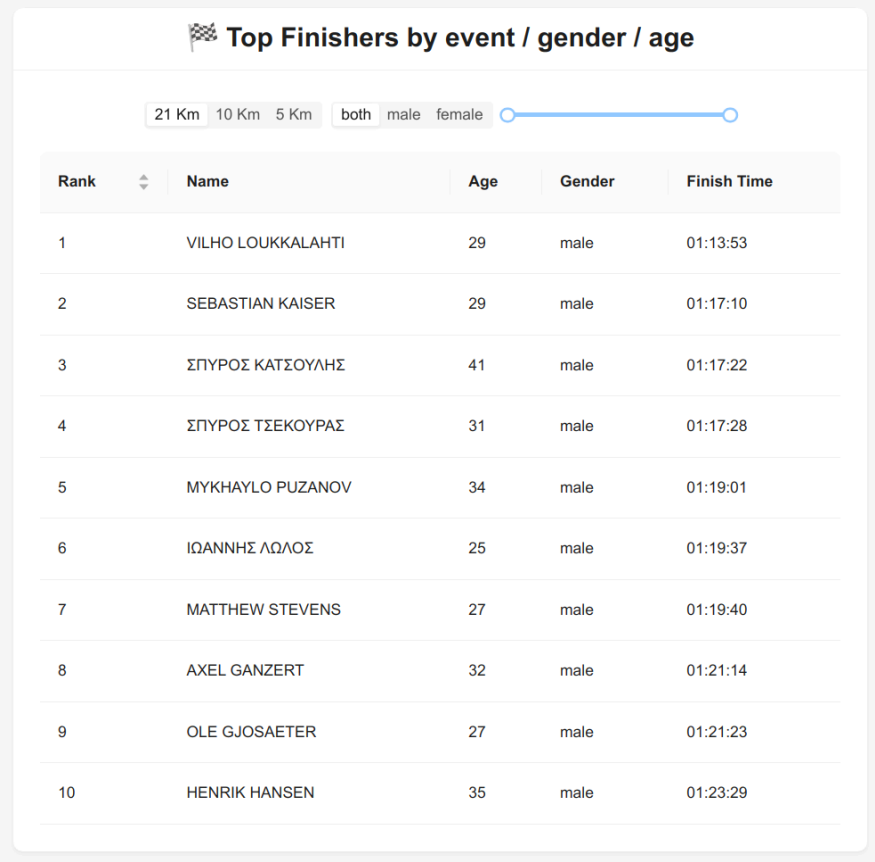


Σχήμα 7: Leaderboard Component Diagram



Σχήμα 8: Leaderboard Sequence Diagram

Ενώ εδώ βλέπουμε το ui του component:



Rank	Name	Age	Gender	Finish Time
1	VILHO LOUKKALAHTI	29	male	01:13:53
2	SEBASTIAN KAISER	29	male	01:17:10
3	ΣΠΥΡΟΣ ΚΑΤΣΟΥΛΗΣ	41	male	01:17:22
4	ΣΠΥΡΟΣ ΤΣΕΚΟΥΡΑΣ	31	male	01:17:28
5	MYKHAYLO PUZANOV	34	male	01:19:01
6	ΙΩΑΝΝΗΣ ΛΩΛΟΣ	25	male	01:19:37
7	MATTHEW STEVENS	27	male	01:19:40
8	AXEL GANZERT	32	male	01:21:14
9	OLE GJOSAETER	27	male	01:21:23
10	HENRIK HANSEN	35	male	01:23:29

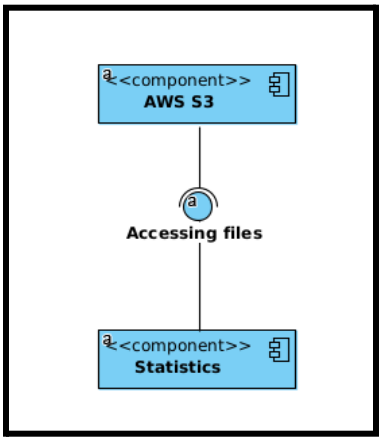
Σχήμα 9: Leaderboard Component UI

Statistics

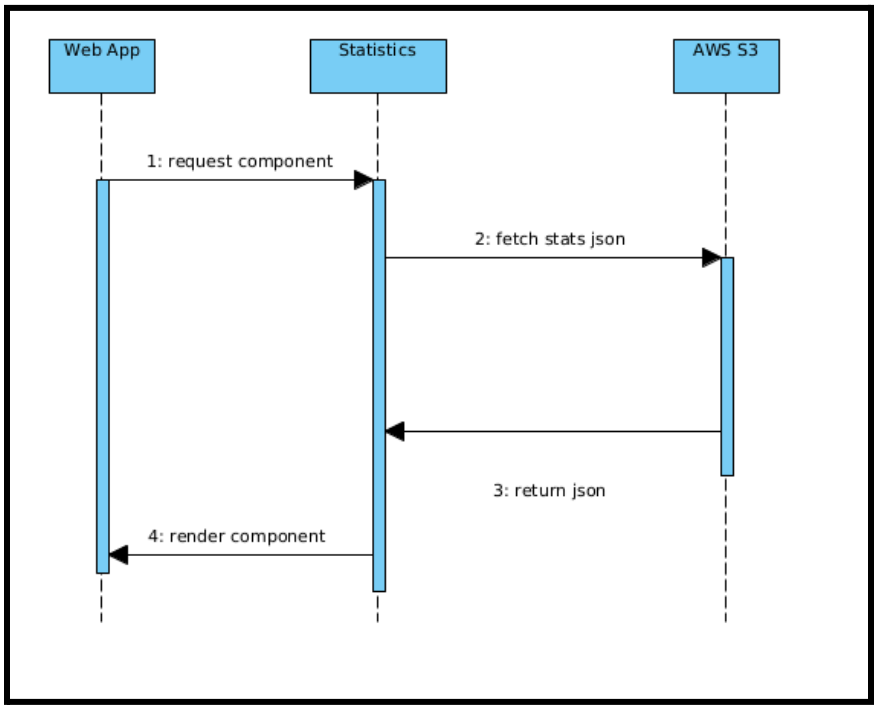
Όπως αναφέραμε έχει προηγηθεί προεπεξεργασία των δεδομένων και δημιουργία ενός json αρχείου στο S3. Κατά το render του component αυτό το json γίνεται load στη μνήμη ώστε να έχουμε διαθέσιμα τα δεδομένα. Χρησιμοποιώ αυτά τα δεδομένα και τα plots του antD ώστε να έχω δύο διαγράμματα, ένα barplot με την ηλικιακή κατανομή των δρομέων ανά event και ένα pie chart με τις πιο δημοφιλείς ομάδες δρομέων σε όλη τη διοργάνωση. Τα δύο αυτά διαγράμματα εμφανίζονται με τη μορφή ενός component με δύο διαφορετικά tabs

.

Τα component και sequence diagrams του στοιχείου:

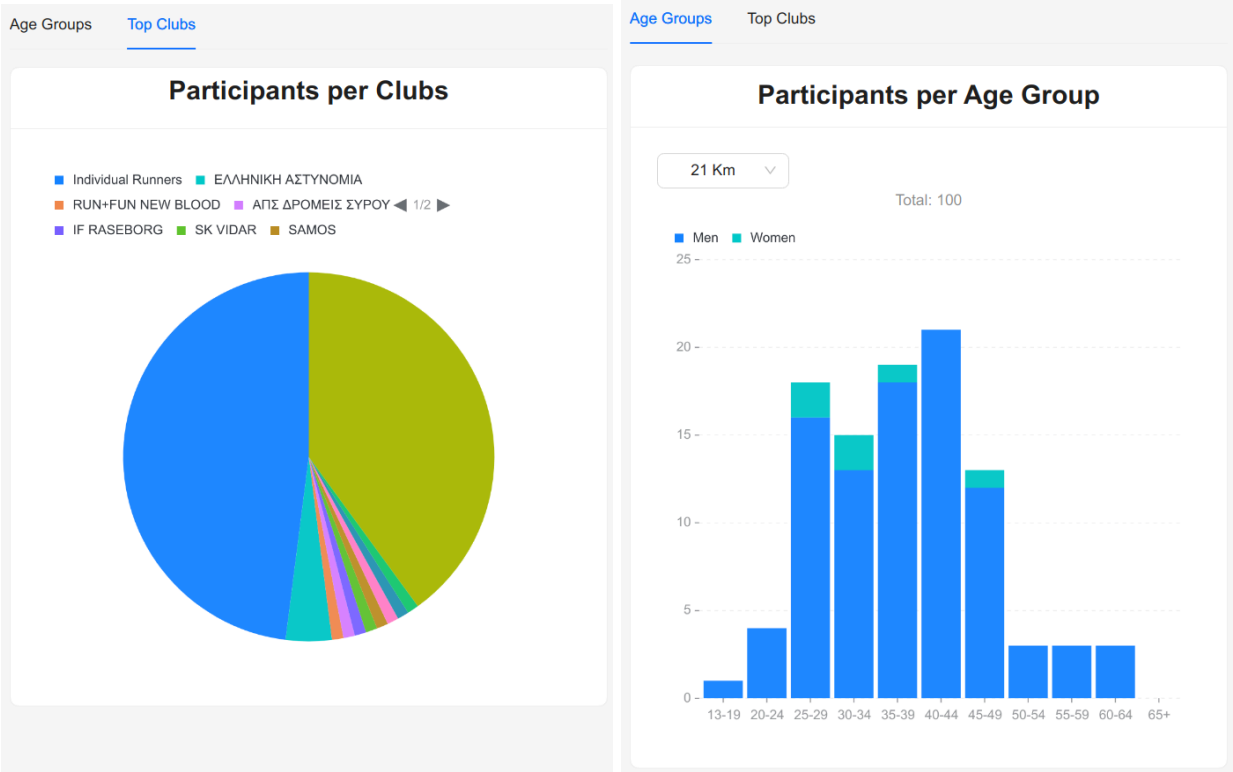


Σχήμα 10: Statistics Component Diagram



Σχήμα 11: Statistics Sequence Diagram

Και το ui του στοιχείου και στα δύο tabs:



Σχήμα 12&13: Statistics Component UI

Runner Search

Για να επιτραπεί το fuzzy-searching, χρησιμοποιώ μία δημοφιλή βιβλιοθήκη που ενδείκνυται για αυτόν τον σκοπό, την Fuse.js .

Λίγα λόγια για την Fuse.js

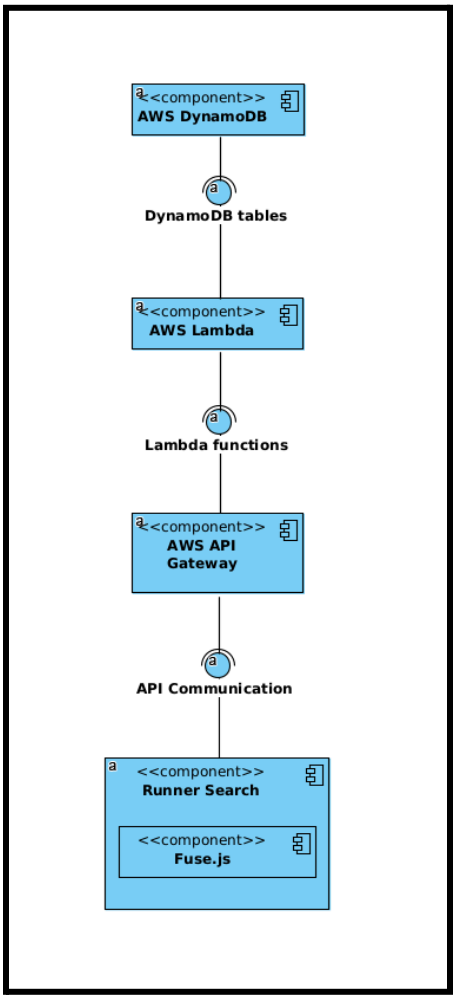
Το **Fuse.js** είναι μια ελαφριά και ισχυρή βιβλιοθήκη JavaScript που επιτρέπει την "fuzzy" αναζήτηση σε σύνολα δεδομένων, δηλαδή αναζήτηση που δεν απαιτεί απόλυτη ταύτιση χαρακτήρων. Είναι ιδιαίτερα χρήσιμη σε εφαρμογές όπου ο χρήστης μπορεί να κάνει ορθογραφικά λάθη ή ασαφείς αναζητήσεις, καθώς βρίσκει αποτελέσματα που μοιάζουν με την αναζητούμενη τιμή.

Η βιβλιοθήκη λειτουργεί στον client ή στον server, είναι χωρίς εξαρτήσεις (zero dependencies), και υποστηρίζει αναζήτηση σε απλά strings ή σε πιο σύνθετα αντικείμενα (π.χ. πίνακες με αντικείμενα που περιέχουν τίτλους, περιγραφές κ.λπ.). Επίσης, επιτρέπει τη ρύθμιση παραμέτρων όπως η ευαισθησία αναζήτησης, τα πεδία στα οποία γίνεται αναζήτηση, και η επιστροφή σχετικότητας των αποτελεσμάτων.

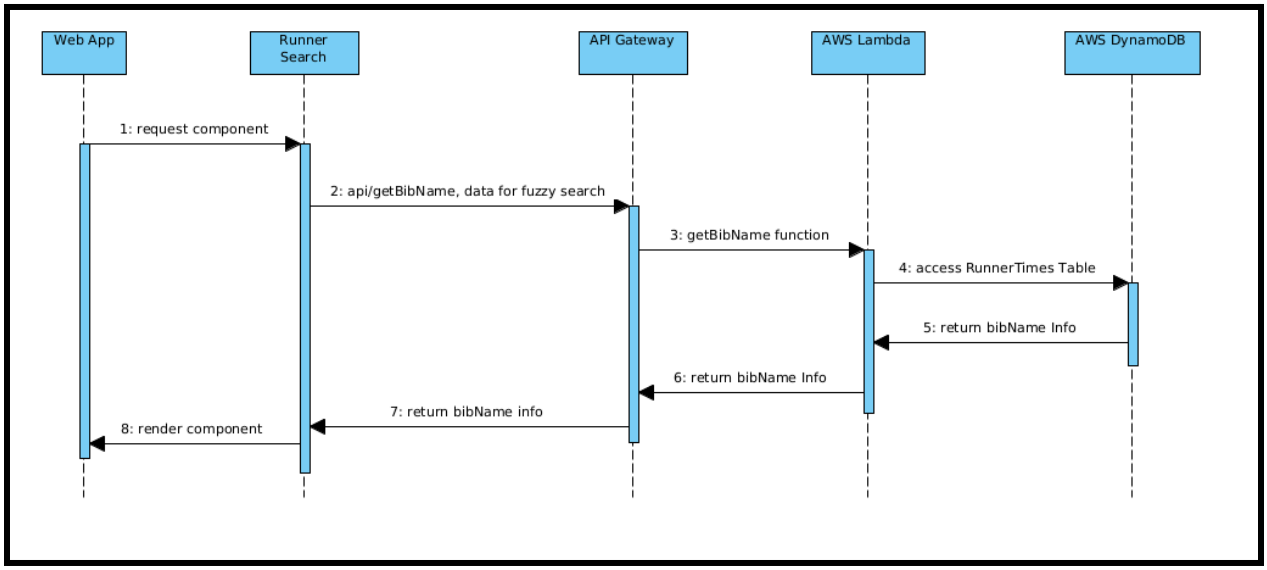
Είναι ιδανική για live search σε React/Next.js εφαρμογές, π.χ. σε φίλτρα λιστών ή dropdowns με autocomplete. [26]

Για να αρχικοποιήσουμε ένα instance μίας fuzzy αναζήτησης με το Fuse.js χρειαζόμαστε ένα σύνολο δεδομένων πάνω στα οποία θα γίνεται η αναζήτηση. Για αυτό δημιουργώ ένα lambda function το οποίο θα διαβάζει το dynamodb table "RunnerTimes" και θα επιστρέφει όλα τα rows του, ωστόσο μόνο τα columns που είναι χρήσιμα σε μία αναζήτηση, δηλαδή το όνομα, το επώνυμο και το bib number. Με αυτά τα δεδομένα προχωρώ στην υλοποίηση του fuzzy searching.
Κάνω deploy το function ξανά με το sam ώστε να είναι διαθέσιμο στο API της εφαρμογής.
Για να υλοποιήσω το component χρησιμοποιώ τα Input και Autocomplete components από το AntD, για μοντέρνη και ευχάριστη εμφάνιση.
Επίσης όταν ο χρήστης επίλεξει κάποιον από τους προτεινόμενους δρομείς, με χρήση του useRouter() hook της Next , φροντίζω ώστε να πηγαίνει στην κατάλληλη σελίδα.

Τα component και sequence diagrams για το component αναζήτησης:



Σχήμα 14:Runner Search Component Diagram



Σχήμα 15: Runner Search Sequence Diagram

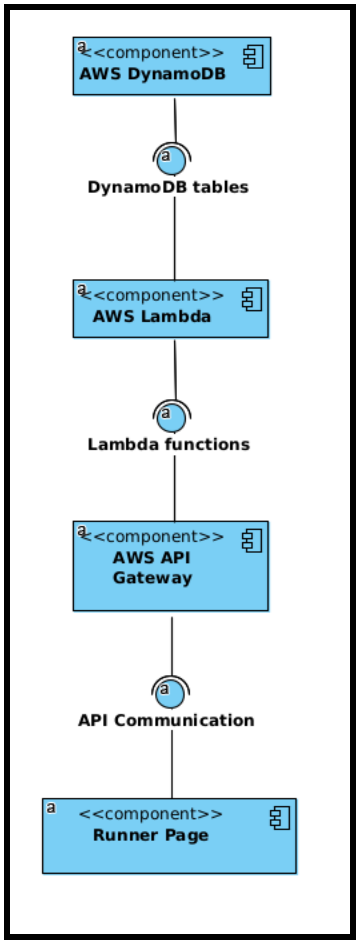
Το τελικό ui του στοιχείου:

Σχήμα 16: Runner Search Component UI

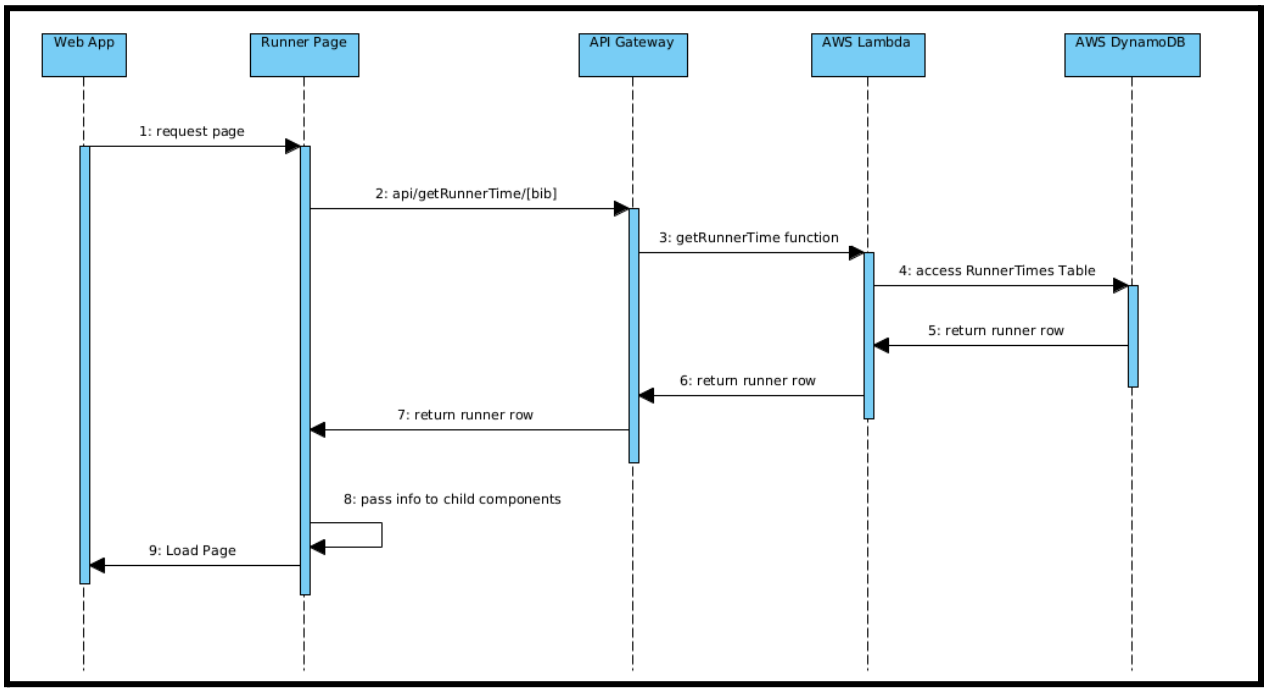
Runner Page

Στη σελίδα αυτή τοποθετούνται τα components AI-Generated Comment, Image Gallery, Runner Info και Race Replay. Η σελίδα παίρνει το bib number από τις παραμέτρους του url και το χρησιμοποιεί για να καλέσει μία lambda function μέσω του API , η οποία φέρνει την πληροφορία του χρήστη από το table “RunnerTimes”. Αυτά τα δεδομένα που λαμβάνει τα δίνει ως είσοδο στα components που τα χρειάζονται ώστε να μη τα φέρνει καθένα ξεχωριστά , αποφεύγοντας περισσότερες κλήσεις στο Amazon Lambda και περισσότερα reads από το dynamodb. Επίσης φροντίζει για την παρουσίαση των δεδομένων και επιτρέπει στον χρήστη να γυρίσει στην κεντρική σελίδα.

Τα component και sequence diagrams για την σελίδα του δρομέα:



Σχήμα 17: Runner Page Component Diagram



Σχήμα 18: Runner Page Sequence Diagram

AI-Generated Comment

Όπως αναφέρθηκε προηγουμένως για αυτό το κομμάτι χρησιμοποιώ την υπηρεσία Amazon Bedrock. Χρησιμοποιώ ένα lambda function, στο οποίο δέχεται στο body ένα json με όλες τις χρήσιμες πληροφορίες του χρήστη:

```
"bib_number",
"lastname",
"firstname",
"fathersname",
"nationality",
"club",
"dateofbirth",
"Split1",
"Split2",
"Split3",
"Split4",
"Split5",
"Split6",
"Split7",
"Split8",
"Finish",
"Finish_realtime",
```

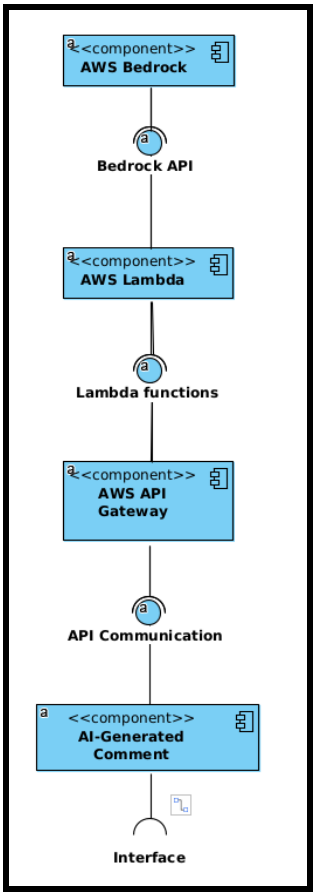
Με αυτό η lambda function, καλεί την invoke_model() του bedrock για το επιλεγμένο μοντέλο με το κατάλληλο request structure, ζητώντας από το llm ένα σύντομο σχόλιο με βάση το json.

```
"messages": [
    {
        "role": "user",
        "content": [
            {
                "type": "text",
                "text": "Below is the information about the performance of a runner in a 21km race: \n "
                + json.dumps(filtered_body)
                + "\n  Make a small comment about their performance as if you are talking to them",
            }
        ],
    }
],
```

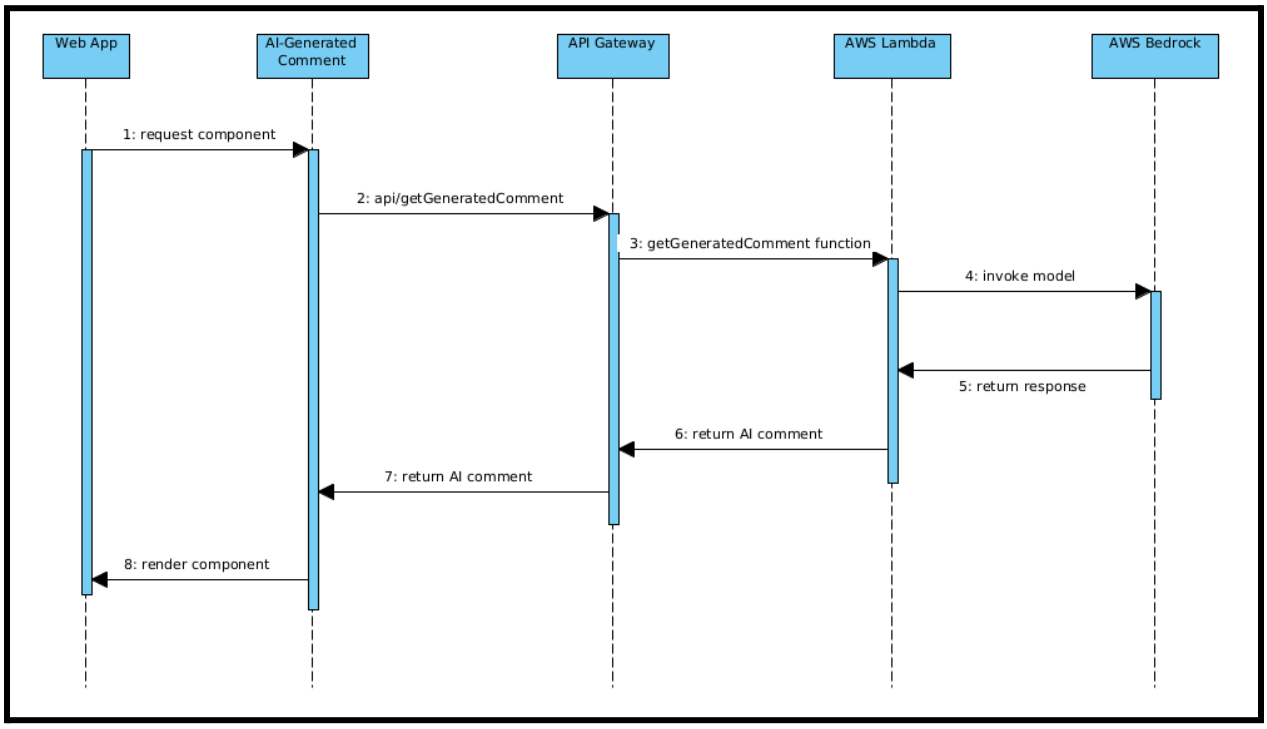
Η lambda function επιστρέφει το κείμενο που έγραψε το llm.
Πάλι γίνεται deploy με το template.yaml του sam και είναι available στο API του API Gateway.

Το react component δέχεται σαν παράμετρο το json με την πληροφορία του χρήστη, η οποία αποκτάται από τον κώδικα στην σελίδα του χρήστη, καθώς επαναχρησιμοποιείται σε άλλα components.
Με την σειρά του το component την περνάει στη lambda function και μετά παρουσιάζει την απάντηση στον χρήστη.

Τα component και sequence diagrams, που αναπαριστούν την παραπάνω διαδικασία:

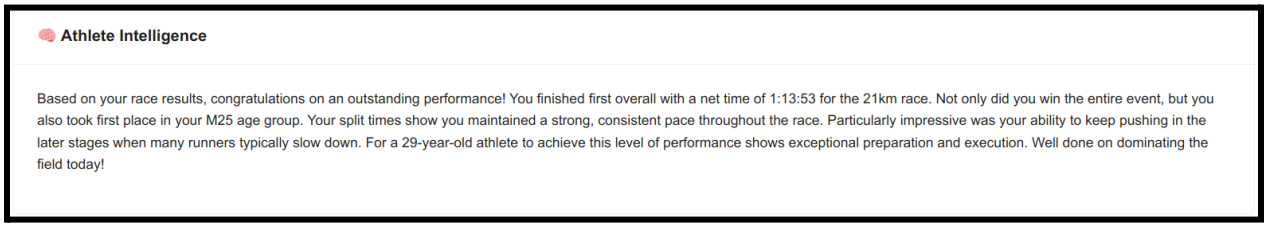


Σχήμα 19: AI-Generated Comment Component Diagram



Σχήμα 20: AI-Generated Comment Sequence Diagram

Και το ui του στοιχείου, όπως προκύπτει:

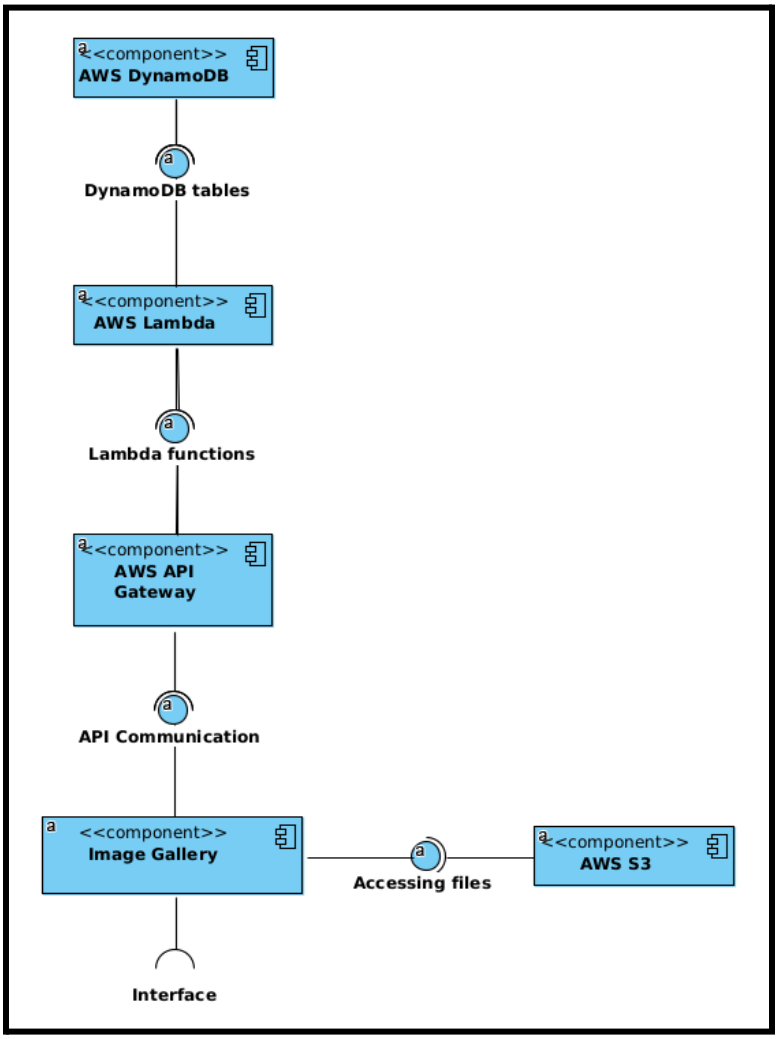


Σχήμα 21: AI-Generated Comment Component UI

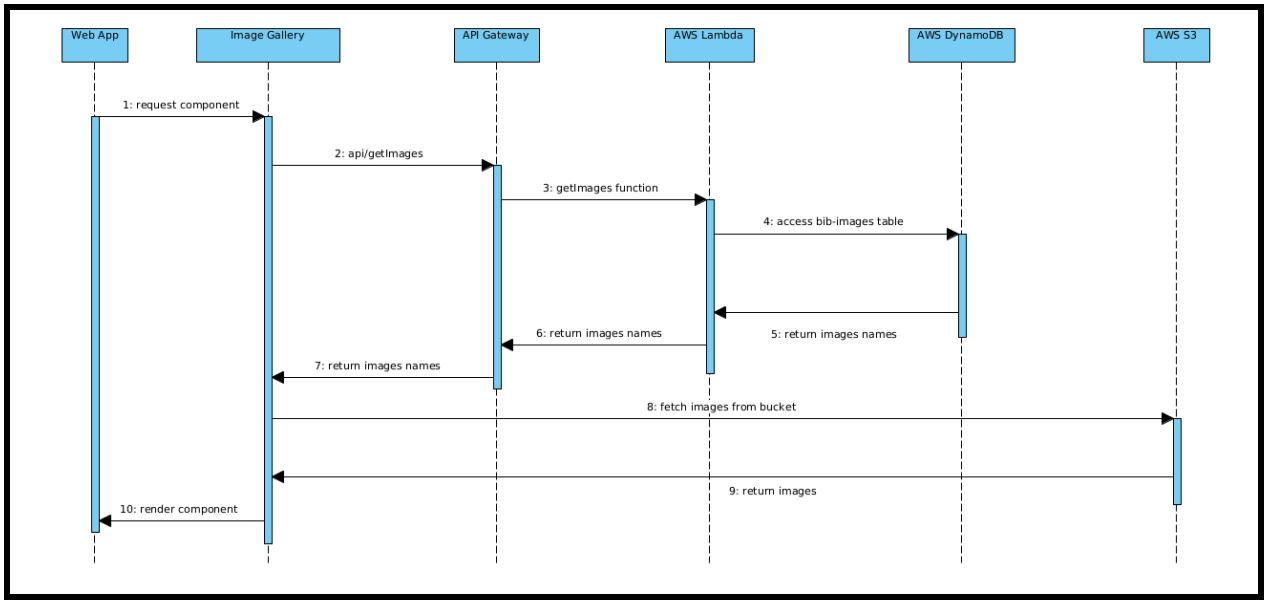
Image Gallery

Για αυτό το component χρειαζόμαστε ένα lambda function το οποίο παίρνει είσοδο το bib number του αντίστοιχου χρήστη και διαβάζει τον πίνακα bib-images2024 του dynamodb , και επιστρέφει τις ονομασίες των κατάλληλων φωτογραφιών στο s3 bucket.
Η συνάρτηση αυτή είναι πάλι διαθέσιμη στο http api στο API Gateway.
Το react component παίρνει το bib number από την σελίδα του δρομέα και με αυτό καλεί τη lambda function. Με τις ονομασίες που αυτή επιστρέφει, το component συνθέτει τα urls του s3 στα οποία είναι διαθέσιμες οι εικόνες για προβολή και τις προβάλλει χρησιμοποιώντας τα components Image και Image.PreviewGroup του AntD, για σύγχρονη και ευχάριστη εμφάνιση.

Τα component και sequence diagrams που αναπαριστούν την παραπάνω διαδικασία:



Σχήμα 22: Image Gallery Component Diagram



Σχήμα 23: Image Gallery Sequence Diagram

Το ui της Image Gallery για έναν δρομέα, όπως φαίνεται αρχικά, και αφού επιλεχθεί μία εικόνα:

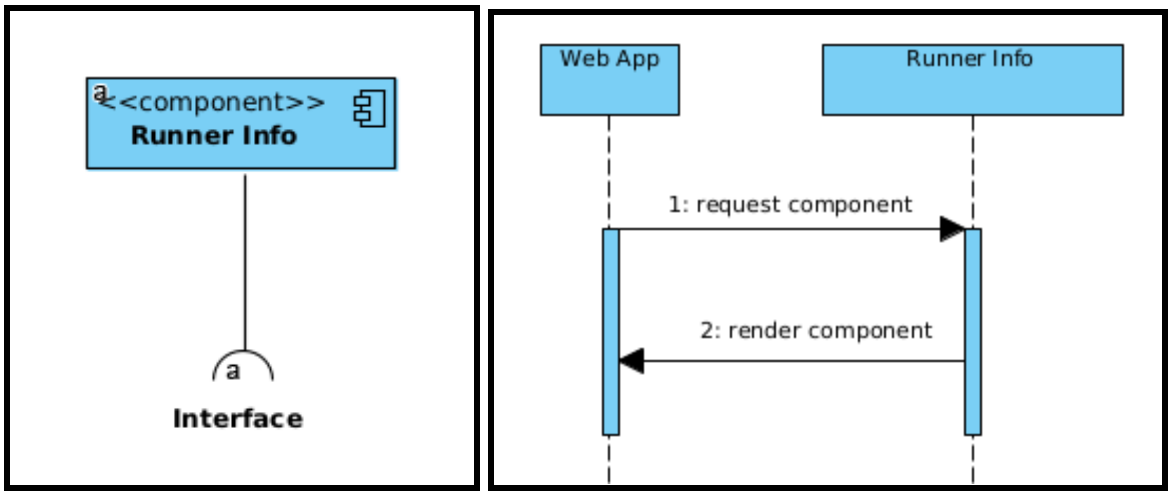
Σχήμα 24: Image Gallery Component UI

Σχήμα 25: Image Gallery Component UI

Runner Info

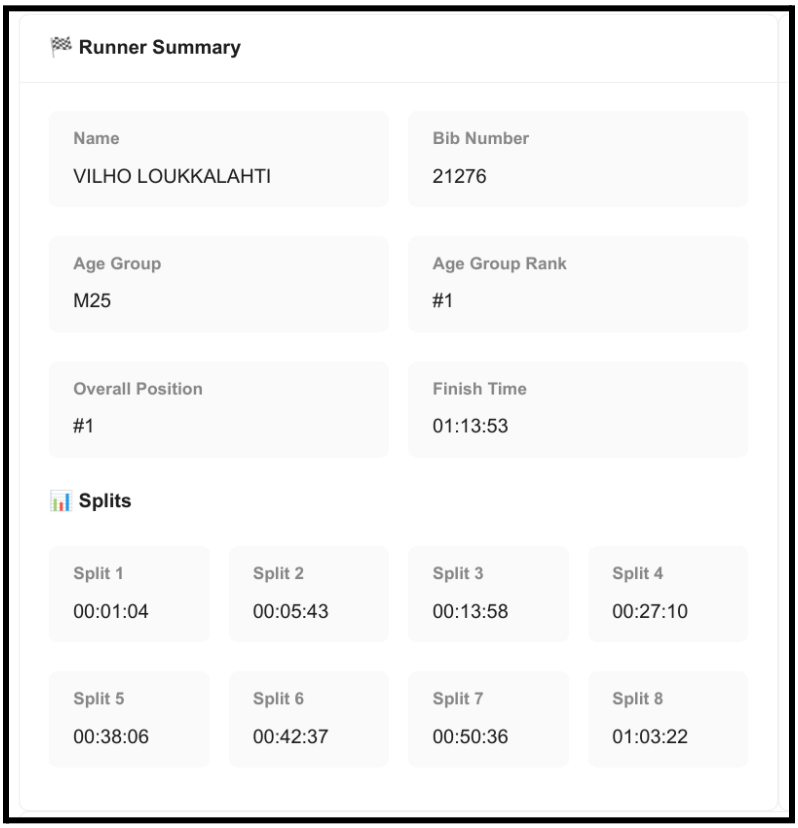
Το component λαμβάνει σαν είσοδο τα δεδομένα από την σελίδα, τα οποία αυτή έχει συλλέξει και έπειτα παρουσιάζει τα αποτελέσματα με φιλικό προς τον χρήστη τρόπο χρησιμοποιώντας τα components του antD.

Αντίστοιχα τα διαγράμματα sequence και component:



Σχήμα 26&27: Runner Info Component Diagram & Sequence Diagram

Και το ui του στοιχείου:



Σχήμα 28: Runner Info Component UI

Race Replay

Για να ενσωματώσω τον χάρτη στην εφαρμογή, πάνω στον οποίο θα φαίνεται το replay του αγώνα με τους δρομείς να κινούνται πάνω στη διαδρομή, χρησιμοποιώ την βιβλιοθήκη MapLibre GL.

> **MapLibre**
> Το MapLibre είναι ένα σύγχρονο, ανοικτού κώδικα οικοσύστημα βιβλιοθηκών για την απεικόνιση διαδραστικών χαρτών σε εφαρμογές ιστού και κινητών συσκευών. Προέκυψε ως fork του Mapbox GL JS όταν αυτό μετατράπηκε σε κλειστού κώδικα, διατηρώντας την υποστήριξη για vector tiles, WebGL rendering και το πρότυπο στυλ Mapbox Style Specification. Η βιβλιοθήκη MapLibre GL JS επιτρέπει την απόδοση χαρτών υψηλής απόδοσης στο web, ενώ το MapLibre Native καλύπτει εφαρμογές Android, iOS και desktop. Το MapLibre υποστηρίζεται από ενεργή κοινότητα και εταιρικούς συνεισφέροντες, παρέχοντας μια ισχυρή και ευέλικτη λύση για ανάπτυξη χαρτογραφικών εφαρμογών χωρίς εξαρτήσεις από κλειστά ή εμπορικά APIs. [27]

Για να απεικονίσω την κίνηση όλων των δρομέων στον χάρτη χρειάζομαι τους χρόνους καθενός από αυτούς. Για αυτό δημιουργώ μία lambda function, η οποία παίρνει ως όρισμα το event (5km, 10km, 21km) στο οποίο έτρεξε ο δρομέας που μας ενδιαφέρει και επιστρέφει τον χρόνο τερματισμού, αλλά και τα ενδιάμεσα splits , κάθε δρομέα που συμμετείχε στο event.
Η συνάρτηση αυτή καλείται από την σελίδα κάθε δρομέα μέσω του API με το event το οποίο απέκτησε η σελίδα όταν έφερε τις πληροφορίες του χρήστη.

Έπειτα συγκεντρώθηκαν οι συντεταγμένες της διαδρομής του αγώνα αλλά και τα σημεία αυτής στα οποία έγιναν οι χρονομετρήσεις των δρομέων.

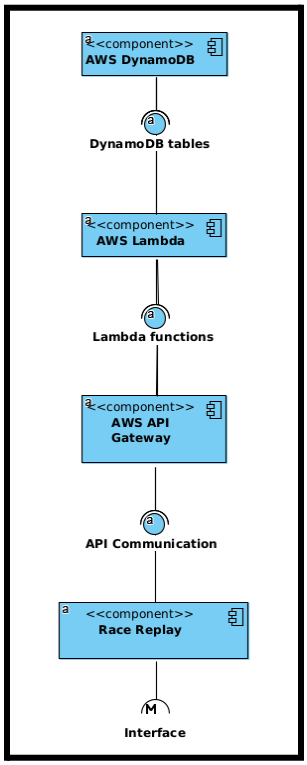
Με βάση τους χρόνους που μετρήθηκαν από δύο διαδοχικά σημεία χρονομέτρησης και της απόστασης μεταξύ τους ακολουθώντας τη διαδρομή του αγώνα, βρίσκεται μία μέση ταχύτητα για κάθε τέτοιο διάστημα. Χρησιμοποιώντας αυτή τη ταχύτητα βρίσκεται, κατά προσέγγιση, η χρονική στιγμή που πέρασε ο δρομέας από κάθε σημείο της διαδρομής.

Για το animation αρχικά τοποθετώ όλους τους δρομείς (ως κουκίδες) στο σημείο εκκίνησης του αγώνα, χρησιμοποιώντας τις συναρτήσεις του MapLibre, με τον δρομέα που μας ενδιαφέρει να αναπαρίσταται με διαφορετικού χρώματος κουκίδα. Χρησιμοποιώντας την συνάρτηση requestAnimationFrame() του browser API, σε κάθε νέο frame που δημιουργεί ο browser, καλώ μία συνάρτηση η οποία αλλάζει τις συντεταγμένες των δρομέων που έχω προσθέσει στον χάρτη, διασφαλίζοντας ομαλή μετακίνηση των κουκίδων μεταξύ των γνωστών σημείων της διαδρομής. Έτσι στον χάρτη που βλέπει ο χρήστης στην οθόνη του , οι δρομείς αλλάζουν θέση. Όταν όλοι οι δρομείς/κουκίδες φτάσουν στο τέλος της διαδρομής το animation σταματάει.

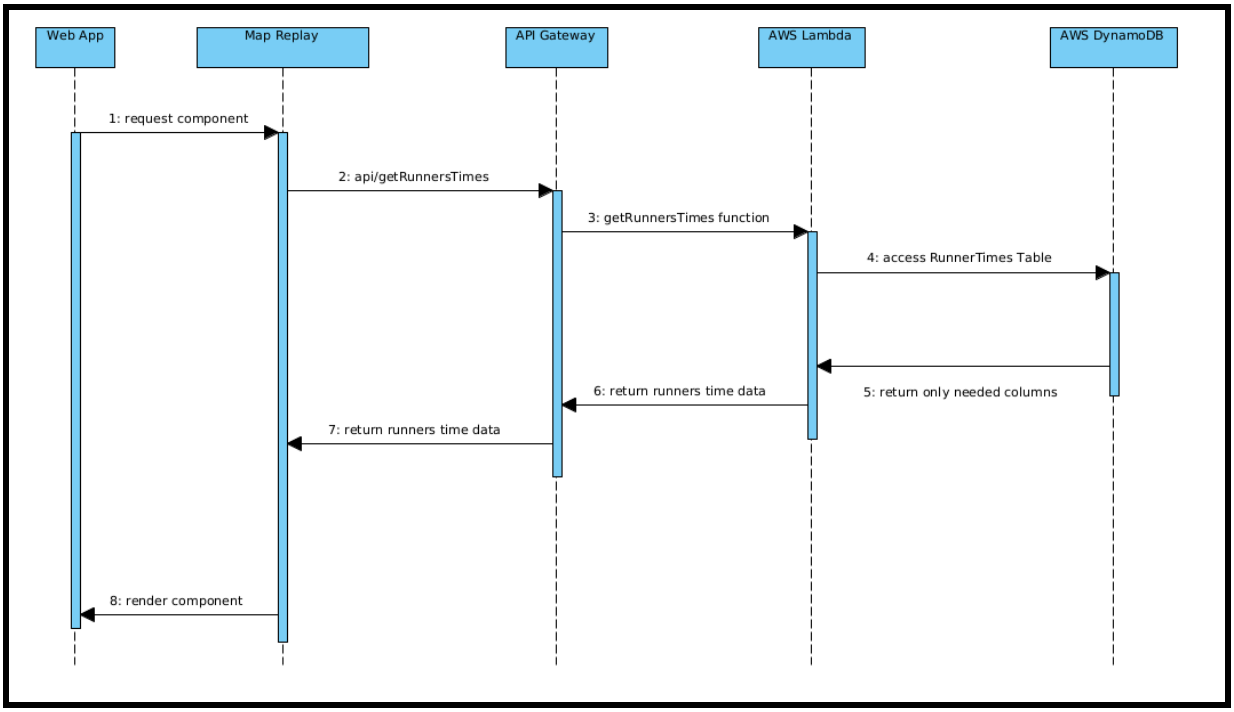
Επίσης χρησιμοποιώντας τα react hooks, useState και useRef, επιτρέπω στον χρήστη να αλλάζει την ταχύτητα του animation, να κάνει start/pause και να βλέπει τον χρόνο στον οποίο βρίσκεται το animation , χωρίς να γίνονται περιττά renders. Για την αλληλεπίδραση με αυτά τα στοιχεία χρησιμοποιώ components του AntD.

Για να είναι όμορφος ο χάρτης, καθώς το MapLibre by default χρησιμοποιεί έναν πολύ απλό χάρτη, διαλέγω ένα πιο ωραίο map style από το Amazon Location Service και θέτω αυτό στον χάρτη του component.

Τα διαγράμματα component και sequence για το component του Race Replay:

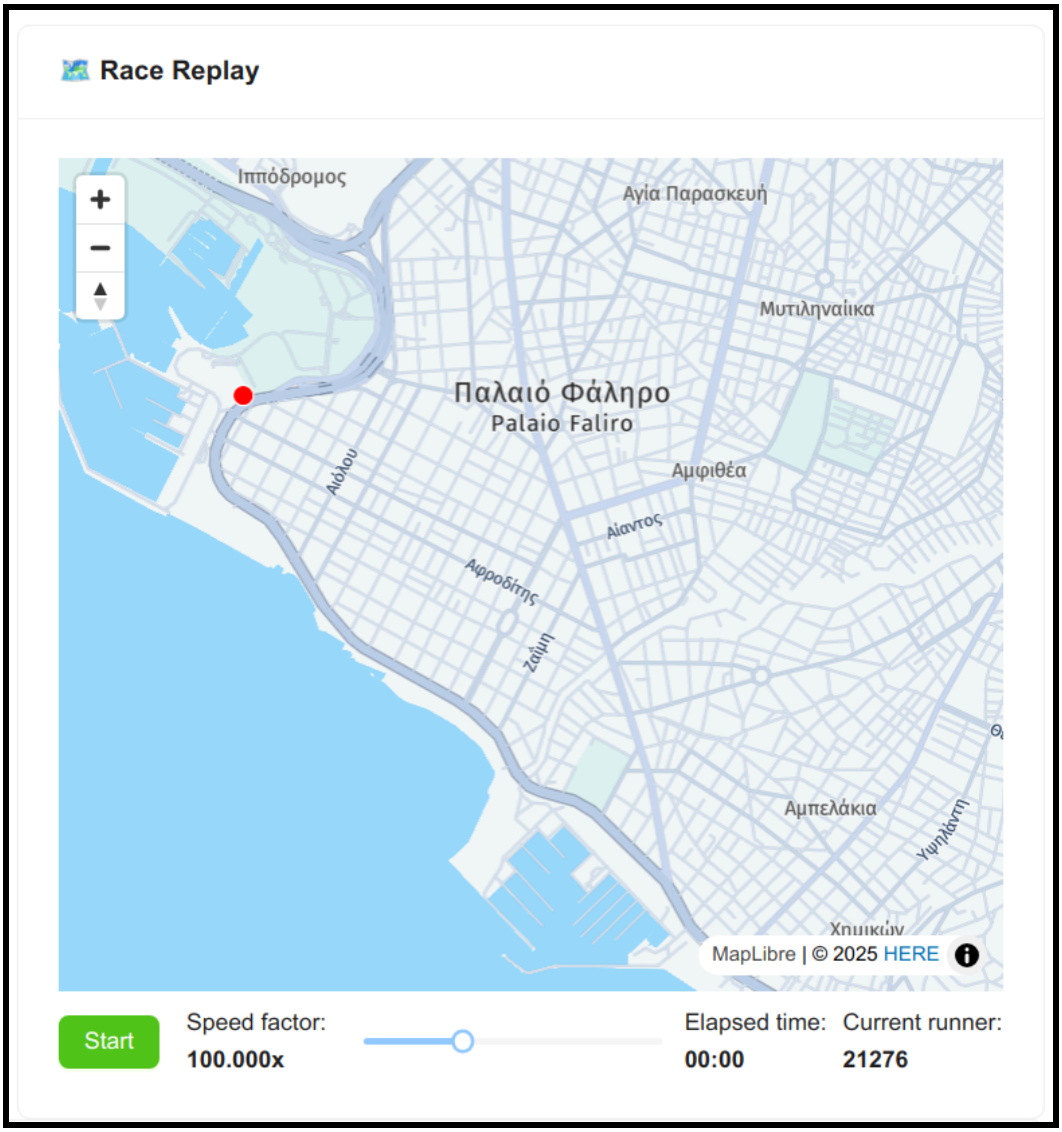


Σχήμα 29: Race Replay Component Diagram



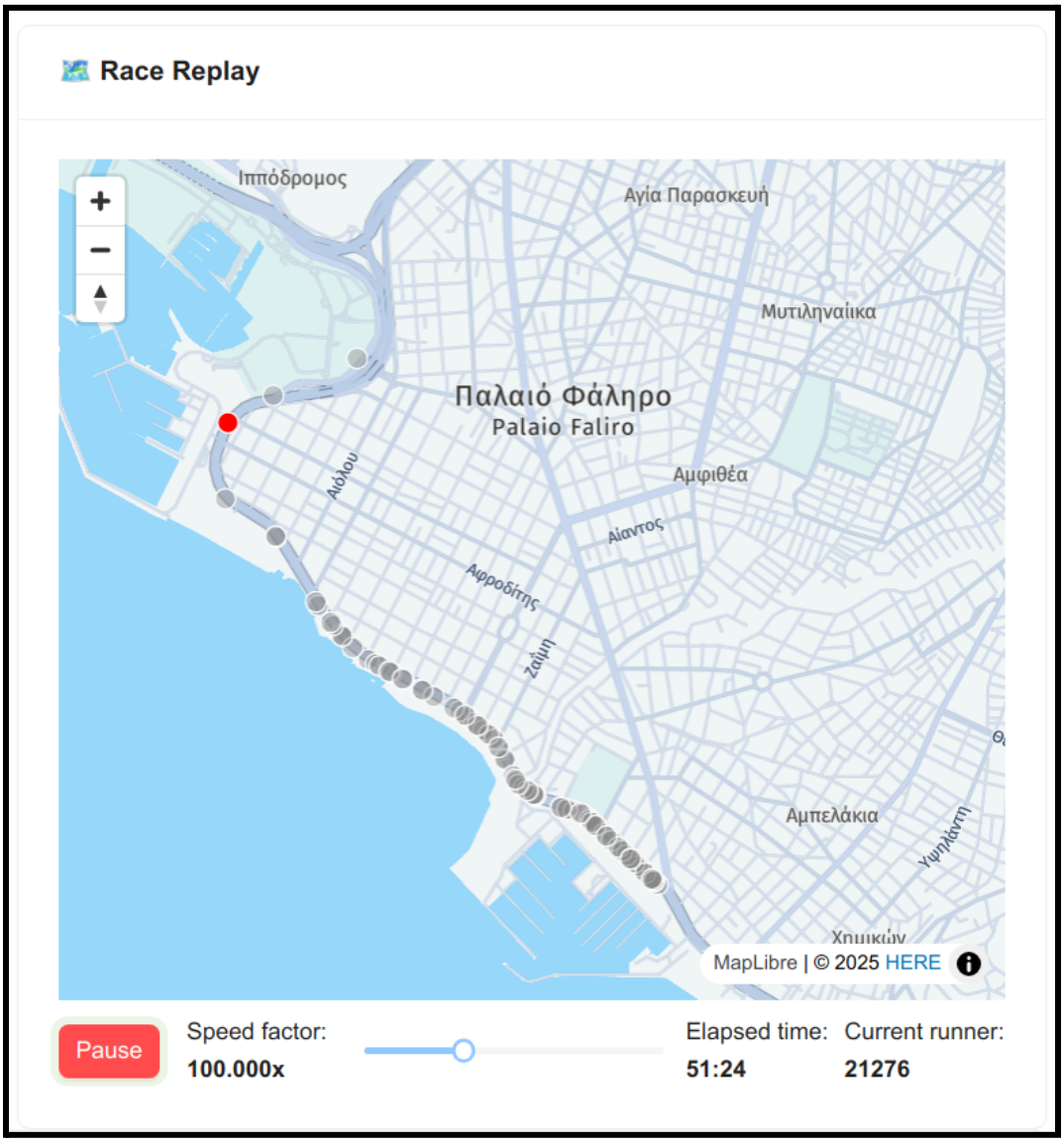
Σχήμα 30: Race Replay Sequence Diagram

Και screenshots του ui πρίν ξεκινήσει το animation:



Σχήμα 31: Race Replay Component UI

Και κατά τη διάρκεια του:

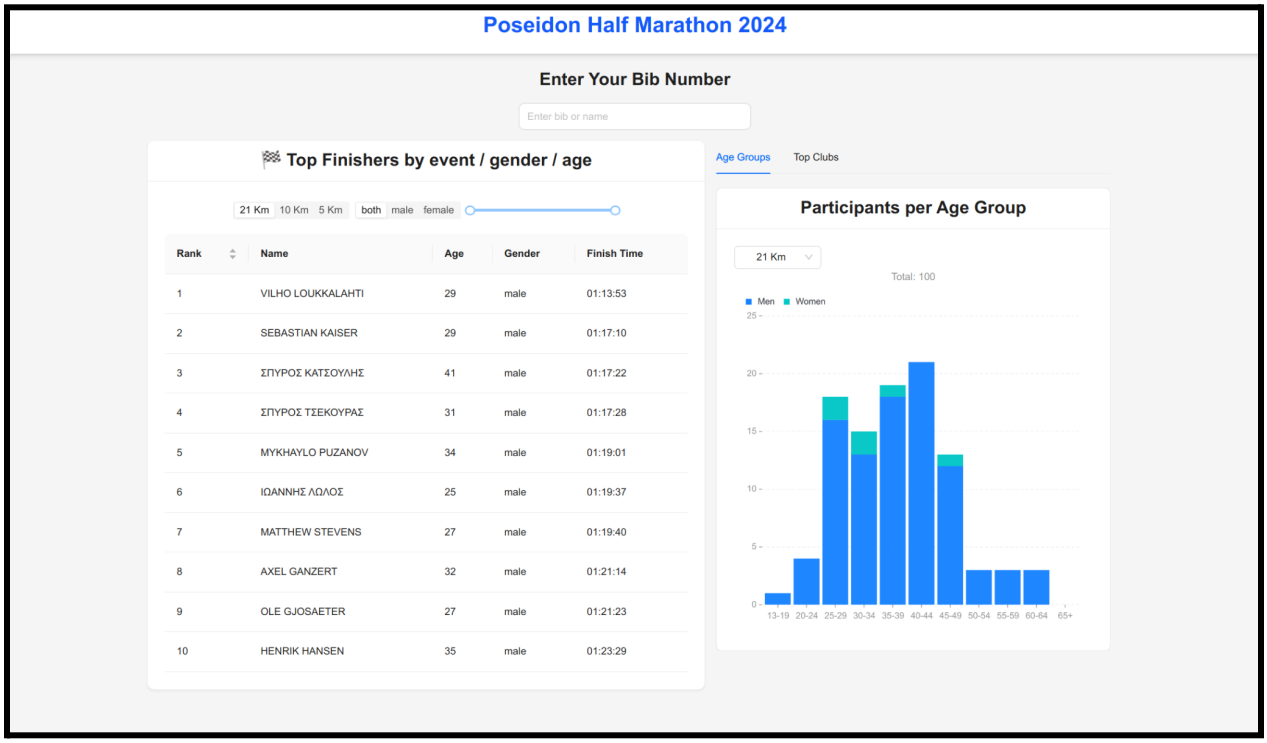


Σχήμα 32: Race Replay Component UI

Κεφάλαιο 4 Εφαρμογή

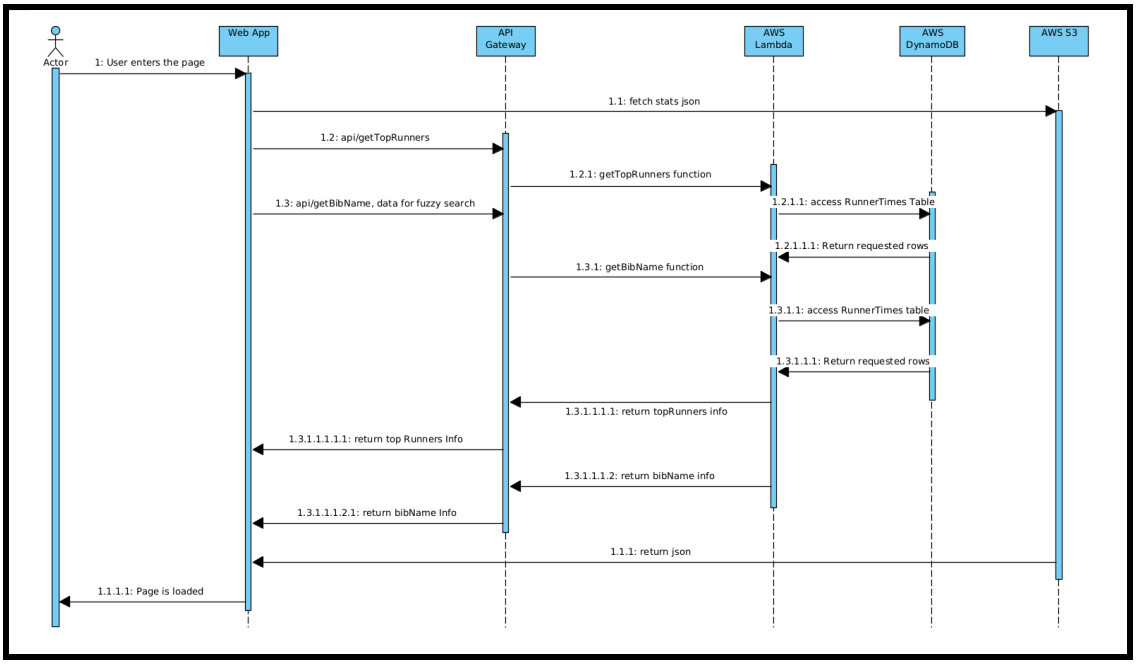
4.1 Η κεντρική σελίδα

Καθώς επισκέπτεται την σελίδα ο χρήστης βλέπει το Leaderboard με τους καλύτερους δρομείς, τα διαγράμματα με στατιστικά για τον αγώνα και το πεδίο αναζήτησης κάποιου δρομέα.
Η κεντρική σελίδα που βλέπει ο χρήστης αρχικά:



Σχήμα 33: Main Page UI

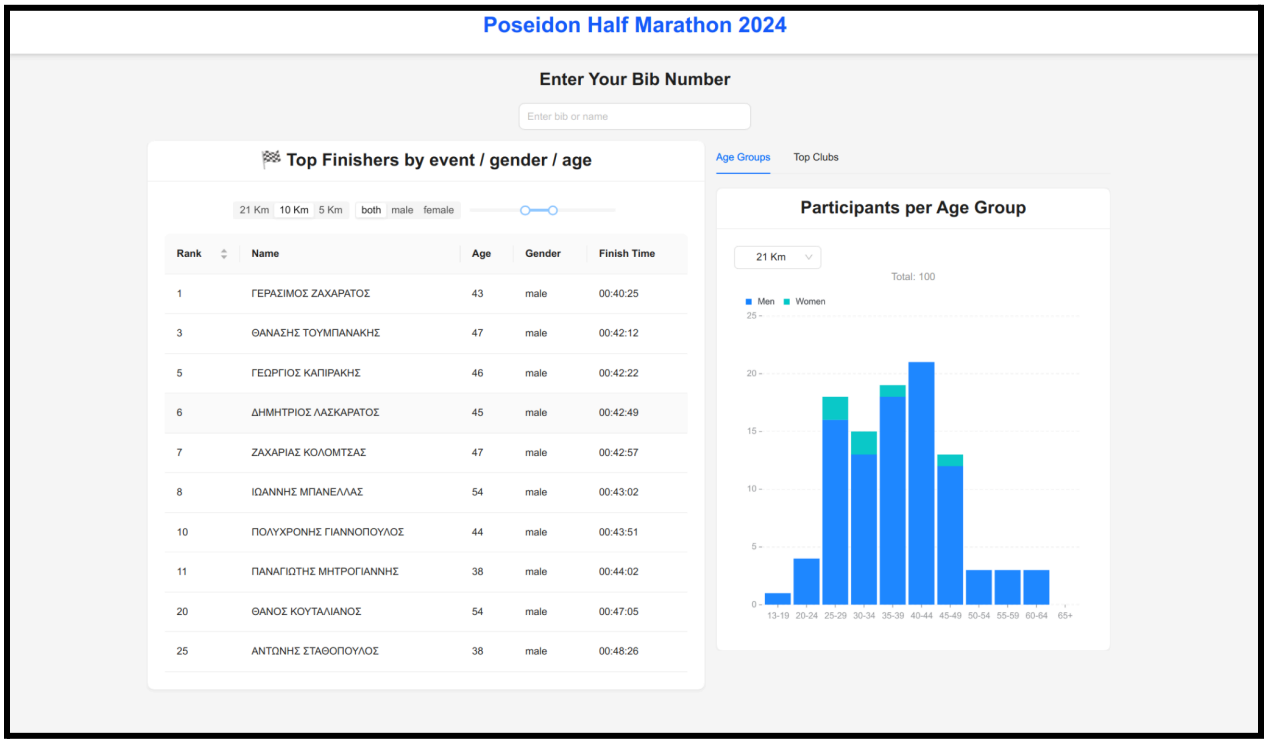
Για να φορτώσει η σελίδα προτού παρουσιαστεί στον χρήστη πρέπει να γίνουν κάποιες ενέργειες. Κάθε component της εφαρμογής, από αυτά που παρουσιάστηκαν στο προηγούμενο κεφάλαιο πρέπει να επικοινωνήσει με τις υπηρεσίες της Amazon ώστε να λάβει τα δεδομένα που χρειάζεται. Η επικοινωνία αυτή φαίνεται στο παρακάτω sequence diagram.
Sequence diagram για την φόρτωση της σελίδας



Σχήμα 34: Main Page Sequence Diagram

Όσο ο χρήστης είναι στη κεντρική σελίδα, ένα από τα πράγματα που μπορεί να κάνει είναι να εφαρμόσει φίλτρα στο leaderboard, ώστε να δει τους καλύτερους δρομείς που ικανοποιούν αυτά τα φίλτρα. Τα φίλτρα είναι σχετικά με το φύλο του δρομέα, το event, την ηλικία και οποιοδήποτε συνδυασμό τους. Παρακάτω φαίνεται ένα παράδειγμα χρήσης. Επιπλέον ο χρήστης μπορεί να επιλέξει έναν δρομέα από την κατάταξη και να μεταβεί απευθείας στη σελίδα του.

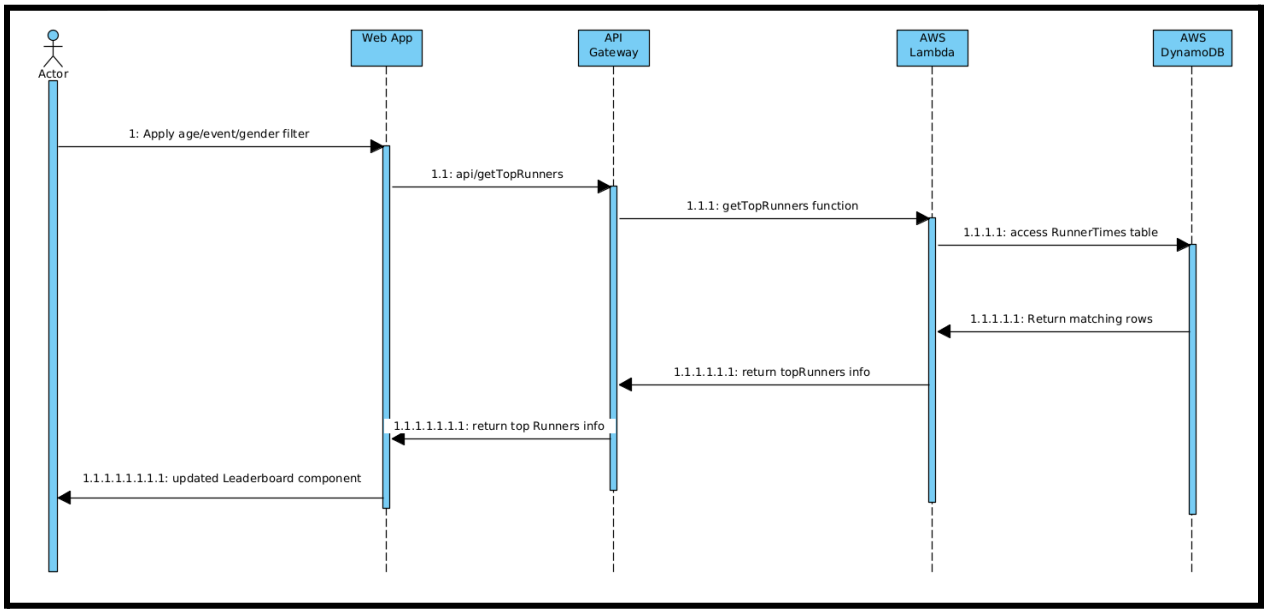
Μετά την εφαρμογή φίλτρων στο leaderboard:



Σχήμα 35: Main Page UI μετά την εφαρμογή φίλτρων

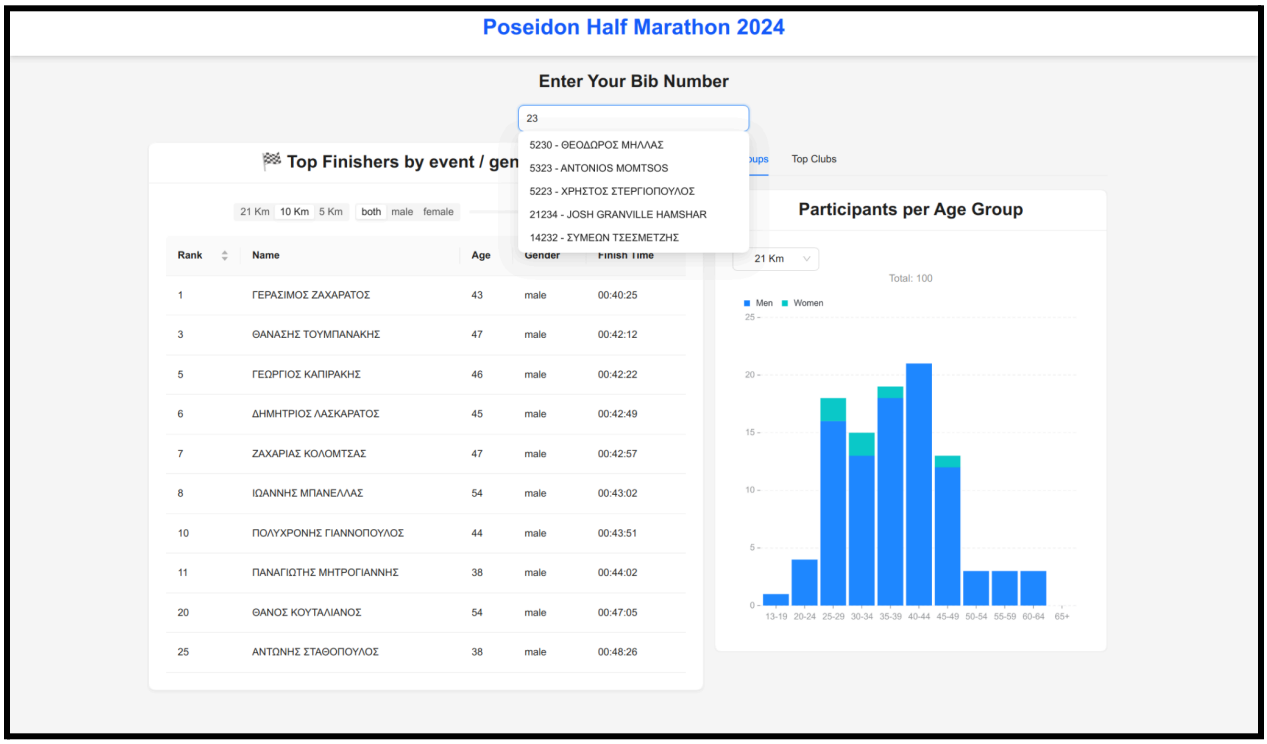
Για να εφαρμοστεί ένα φίλτρο πρέπει το component του leaderboard να κάνει εκ νέου ένα request ώστε να φέρει τους καλύτερους αθλητές που ικανοποιούν τα νέα φίλτρα, η διαδικασία φαίνεται στο παρακάτω διάγραμμα.

Sequence diagram για την εφαρμογή φίλτρων:



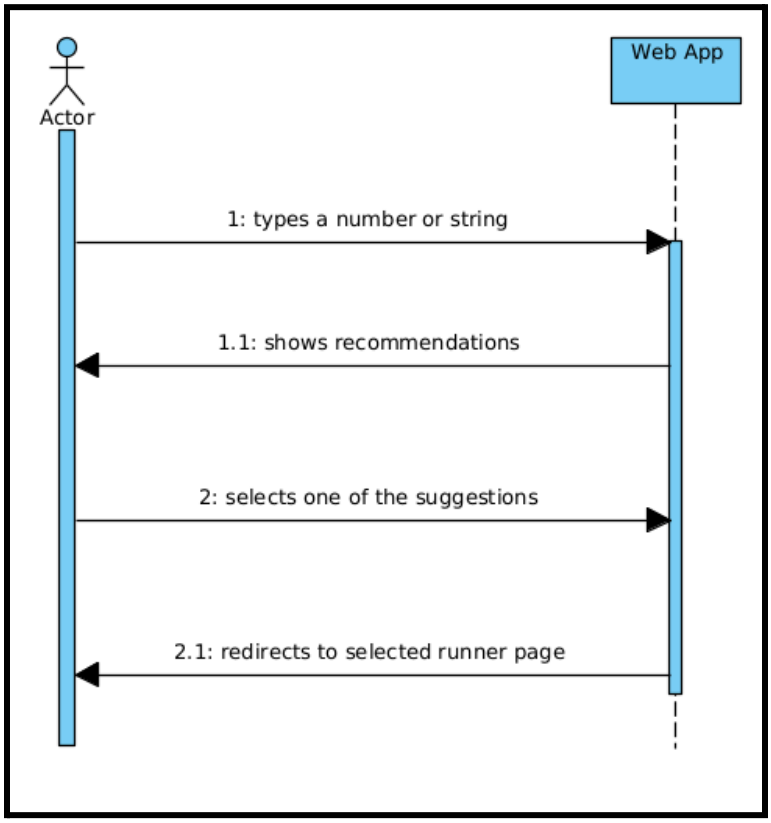
Σχήμα 36: Sequence Diagram για εφαρμογή φίλτρων στο Leaderboard

Η βασική λειτουργία της κεντρικής σελίδας είναι να επιτρέπει την αναζήτηση κάποιου συγκεκριμένου δρομέα και να μεταβαίνει στη σελίδα του. Μέσω του searchBar η αναζήτηση μπορεί να γίνει είτε με το bib number ή με το όνομα του δρομέα στη περίπτωση που δε θυμάται το bib number. Καθώς ο χρήστης πληκτρολογεί εμφανίζονται προτάσεις με βάση αυτό που έχει ήδη πληκτρολογήσει, ενώ υποστηρίζεται και fuzzy searching. Όταν ο χρήστης επιλέξει μία από τις προτεινόμενες επιλογές , πηγαίνει στην αντίστοιχη σελίδα. Παρακάτω φαίνεται η εφαρμογή κατά την αναζήτηση.
Αναζήτηση κάποιου δρομέα:



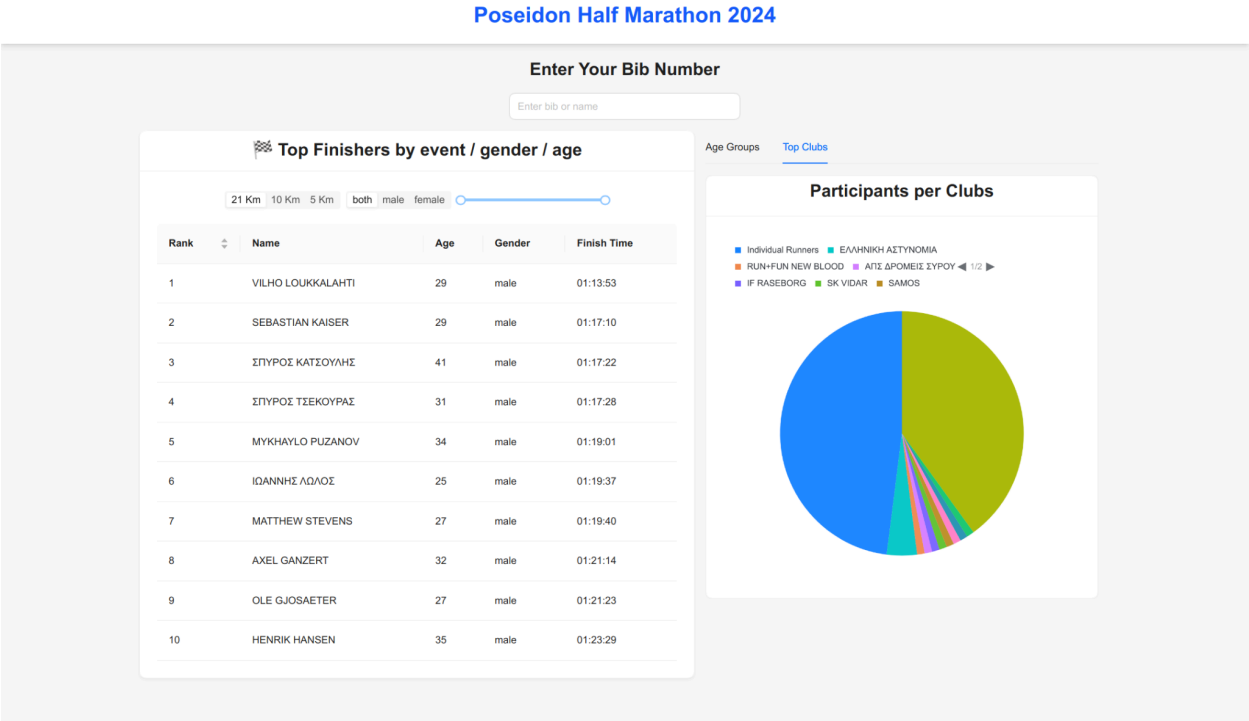
Σχήμα 37: Main Page UI κατά την αναζήτηση δρομέα

Τα δεδομένα που χρησιμοποιούνται στην αναζήτηση έχουν αποκτηθεί ήδη από την φόρτωση της σελίδας, οπότε η αναζήτηση γίνεται εξ ολοκλήρου στο frontend κομμάτι της εφαρμογής, όπως φαίνεται στο παρακάτω διάγραμμα.
Sequence diagram κατά την αναζήτηση:



Σχήμα 37: Sequence Diagram για την αναζήτηση δρομέα

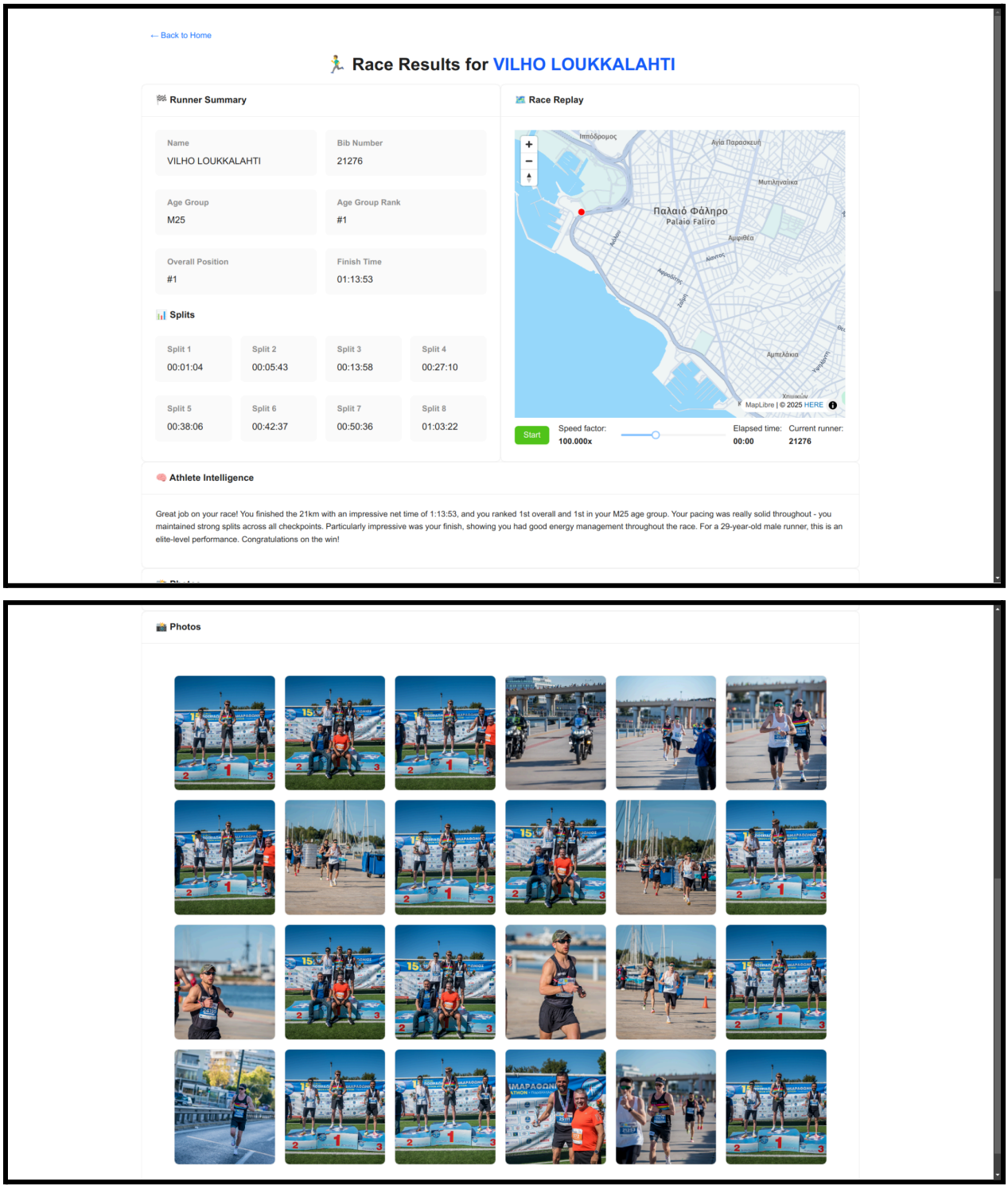
Ο χρήστης επίσης επιλέγοντας το tab Top Clubs βλέπει το δεύτερο διάγραμμα με τα στατιστικά του αγώνα, την κατανομή με τα δημοφιλέστερα clubs.



Σχήμα 38: Main Page UI δεύτερο tab Statistics

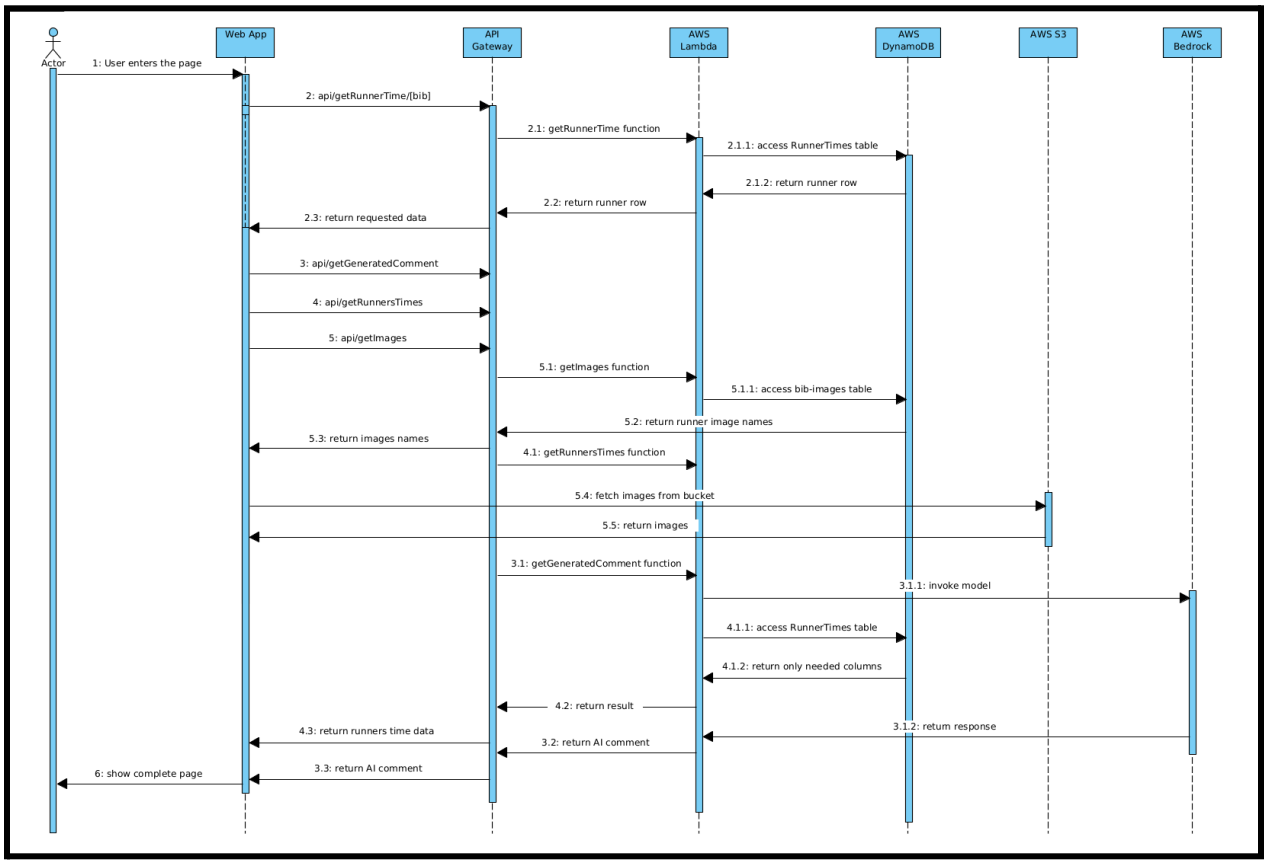
Η σελίδα του δρομέα

Στη σελίδα του δρομέα εμφανίζονται οι πληροφορίες σχετικά με τον δρομέα και την απόδοση του. Ακόμα ο χάρτης με το replay της διαδρομής, το AI-Generated comment και οι φωτογραφίες του δρομέα όπως ανιχνεύτηκαν από το AI σύστημα. Το ui:



Σχήμα 39&40: Runner Page UI

Για να εμφανιστεί η παραπάνω πληροφορία πρέπει τα components της σελίδας του δρομέα να επικοινωνήσουν με τις υπηρεσίες της Amazon προκειμένου να συγκεντρωθεί όλη η απαραίτητη πληροφορία στο frontend. Η διαδικασία αυτή φαίνεται στο παρακάτω διάγραμμα.
Το sequence diagram για την φόρτωση της σελίδας:



Σχήμα 41: Runner Page Sequence Diagram

Κεφάλαιο 5 Επίλογος/Μελλοντικές Προσθήκες

Η εφαρμογή προς το παρόν εκπληρώνει τον βασικό στόχο μιας εφαρμογής για αγώνες τρεξίματος, μαζί με μερικές πρωτότυπες λειτουργίες. Ωστόσο υπάρχουν επιπλέον πράγματα τα οποία μπορούν να γίνουν για να αναβαθμιστεί η εφαρμογή ακόμα παραπάνω. Η υλοποίηση τους ξεπερνά τα πλαίσια αυτής της διπλωματικής , οπότε απλά τα αναφέρω.

5.1 Ανανέωση της σελίδας σε ζωντανό χρόνο καθώς εξελίσσεται ο αγώνας

Μία πιθανή επέκταση είναι η αναβάθμιση της εφαρμογής ώστε όπως εξελίσσεται ο αγώνας, οι μετρήσεις που γίνονται στα checkpoints να περνιούνται αυτόματα στο table με τους χρόνους των δρομέων. Με αυτόν τον τρόπο οι φίλοι και συγγενείς των δρομέων θα μπορούν να έχουν μία ζωντανή εικόνα για το πως τα πηγαίνουν οι δικοί τους στον αγώνα. Ακόμα οι δρομείς θα έχουν τα αποτελέσματα απευθείας διαθέσιμα μετά τη λήξη του αγώνα χωρίς να χρειάζεται να γίνει κάποια ενέργεια, και συνεπώς να χαθεί χρόνος, από τους διοργανωτές για το πέρασμα των αποτελεσμάτων στην εφαρμογή.

Πέρα από τους χρόνους, αυτή η λειτουργικότητα θα ήταν χρήσιμη και για τις φωτογραφίες. Θα έπρεπε να δημιουργηθεί κάποιο σύστημα που όπως οι φωτογράφοι βγάζουν μία φωτογραφία, αυτή να περνάει από το σύστημα κατηγοριοποίησης και να εμφανίζεται στη σελίδα του κατάλληλου δρομέα.

5.2 Σχόλια και Likes στις σελίδες των αθλητών

Μία ακόμη ενδιαφέρουσα προσθήκη θα ήταν η δρομείς, ή οι φίλοι και συγγενείς τους, να μπορούν να δίνουν likes ή να κάνουν comments στην σελίδα των υπολοίπων δρομέων (ή και του εαυτού τους) . Με αυτή τη προσθήκη οι χρήστες μπορούν να δείχνουν τον θαυμασμό και την επιβράβευση τους προς τους άλλους δρομείς, και να τους ενθαρρύνουν. Έτσι η εφαρμογή αποκτά έναν πιο κοινωνικό χαρακτήρα και βελτιώνει την εμπειρία των συμμετεχόντων στο event.

Για να υλοποιηθεί το παραπάνω θα χρειαζόμασταν λογαριασμούς για τους χρήστες και κάποιο σύστημα authentication και ελέγχου. Αυτό για να είναι σαφές ποιος έκανε ποιο σχόλιο αλλά και για να φιλτράρονται πολύ αρνητικά/προσβλητικά σχόλια.

5.3 Δημιουργία προσωποποιημένου collage/award μετά τον αγώνα

Πολλοί αθλητές που συμμετέχουν σε τέτοια events, μετά τον τερματισμό τους, νιώθουν περήφανοι για τον εαυτό τους και μοιράζονται το επίτευγμα τους στα μέσα κοινωνικής δικτύωσης, συνήθως με την μορφή φωτογραφιών. Επομένως θα ήταν χρήσιμο, με τα δεδομένα της απόδοσης του δρομέα, αλλά και μερικές από τις φωτογραφίες του, να δημιουργείται ένα poster/collage με αυτή τη πληροφορία, το οποίο να είναι σε μορφή, έτοιμη για κοινοποίηση στα social media. Με μία τέτοια λειτουργία, η διοργάνωση μπορεί να λάβει περισσότερη έκθεση στα social media, και έτσι να οδηγηθεί σε μεγαλύτερο αριθμό συμμετεχόντων στα επόμενα events.
Το παραπάνω μπορεί να εμπλουτιστεί με κάποια χρήση της τεχνητής νοημοσύνης, ώστε να μη φαίνεται τόσο αυτοματοποιημένο και να μοιάζει πιο προσωποποιημένο για τους χρήστες.

Κλείνοντας, αξίζει να αναφερθεί ότι οι πόροι για την χρήση των υπηρεσιών της Amazon παραχωρήθηκαν με την υποστήριξη του GRNET.

Βιβλιογραφία

[1] Anthropic, “Claude 3.7 Sonnet and Claude Code,” *Anthropic.com*, 2025. https://www.anthropic.com/news/claude-3-7-sonnet

[2] Meta, “Llama 3.2: Revolutionizing edge AI and vision with open, customizable models,” *Meta.com*, 2022. https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/

[3] “Pixtral Large | Mistral AI,” *Mistral.ai*, 2024. https://mistral.ai/news/pixtral-large

[4] “Amazon Nova Understanding Models,” *Amazon Web Services, Inc.*, Sep. 02, 2025. https://aws.amazon.com/ai/generative-ai/nova/understanding/ (accessed Feb. 05, 2025).

[5] “Amazon Titan Text models - Amazon Bedrock,” *docs.aws.amazon.com*. https://docs.aws.amazon.com/bedrock/latest/userguide/titan-text-models.html

[6] I. Ben-Ami, T. Basha, and S. Avidan, “Racing Bib Number Recognition.” Available: https://people.csail.mit.edu/talidekel/papers/RBNR.pdf

[7] M. Castrillón-Santana *et al.*, “An Evaluation of General-Purpose Optical Character Recognizers and Digit Detectors for Race Bib Number Recognition,” *Proceedings of the 13th International Conference on Pattern Recognition Applications and Methods*, pp. 910–917, 2024, doi: https://doi.org/10.5220/0012562400003654.

[8] A. Apap and D. Seychell, “Marathon Bib Number Recognition using Deep Learning,” *2019 11th International Symposium on Image and Signal Processing and Analysis (ISPA)*, pp. 21–26, Sep. 2019, doi: https://doi.org/10.1109/ISPA.2019.8868768.

[9] X. Wang and J. Yang, “Marathon athletes number recognition model with compound deep neural network,” *Signal, Image and Video Processing*, vol. 14, no. 7, pp. 1379–1386, Apr. 2020, doi: https://doi.org/10.1007/s11760-020-01677-5.

[10] K. Varshney, “Marathon Bib Identification and Recognition,” *Medium*, Feb. 16, 2019. https://medium.com/data-science/marathon-bib-identification-and-recognition-25ee7e08d118 (accessed Feb. 05, 2025).

[11] AWS, “Amazon Rekognition – Video and Image - AWS,” *Amazon Web Services, Inc.*, 2017. https://aws.amazon.com/rekognition/

[12] “InsightFace,” *insightface.ai*. https://insightface.ai/

[13] Microsoft, “TypeScript - JavaScript that scales.,” *Typescriptlang.org*, 2025. https://www.typescriptlang.org/

[14] PYTHON, “Python,” *Python.org*, 2025. https://www.python.org/

[15] Meta Open Source, “React,” *react.dev*, 2025. https://react.dev/

[16] Vercel, “Next.js by Vercel - The React Framework,” *nextjs.org*, 2024. https://nextjs.org/

[17] “Ant Design - A UI Design Language and React UI library,” *Ant.design*, 2019. https://ant.design/

[18] tailwindcss, “Tailwind CSS - Rapidly build modern websites without ever leaving your HTML.,” *tailwindcss.com*, 2025. https://tailwindcss.com/

[19] AWS, “Cloud Object Storage | Store & Retrieve Data Anywhere | Amazon Simple Storage Service,” *Amazon Web Services, Inc.*, 2025. https://aws.amazon.com/s3/

[20] Amazon Web Services, “Amazon DynamoDB - Overview,” *Amazon Web Services, Inc.*, 2024. https://aws.amazon.com/dynamodb/

[21] AWS, “AWS Lambda – Serverless Compute,” *Amazon Web Services, Inc.*, 2019. https://aws.amazon.com/lambda/

[22] “AWS Serverless Application Model - Amazon Web Services,” *Amazon Web Services, Inc.* https://aws.amazon.com/serverless/sam/

[23] AWS, “AWS Amplify | Scalable, Full-Stack Web & Mobile Apps | Amazon Web Services,” *Amazon Web Services, Inc.*, 2024. https://aws.amazon.com/amplify/

[24] AWS, “Foundation Model API Service – Amazon Bedrock – AWS,” *Amazon Web Services, Inc.*, 2025. https://aws.amazon.com/bedrock/

[25] aws, “Amazon API Gateway,” *Amazon Web Services, Inc.*, 2019. https://aws.amazon.com/api-gateway/

[26] “Fuse.js | Fuse.js,” *Fusejs.io*, 2024. https://www.fusejs.io/

[27] map, “MapLibre,” *maplibre.org*. https://maplibre.org/

[28] C. M. Bishop, *ΑΝΑΓΝΩΡΙΣΗ ΠΡΟΤΥΠΩΝ ΚΑΙ ΜΗΧΑΝΙΚΗ ΜΑΘΗΣΗ*. ΓΡΗΓΟΡΙΟΣ ΧΡΥΣΟΣΤΟΜΟΥ ΦΟΥΝΤΑΣ, 2019.

[29] Ν. ΑΒΟΥΡΗΣ, Ν. ΤΣΕΛΙΟΣ, Χ. ΚΑΤΣΑΝΟΣ, and Κ. ΜΟΥΣΤΑΚΑΣ, *ΕΙΣΑΓΩΓΗ ΣΤΗΝ ΑΛΛΗΛΕΠΙΔΡΑΣΗ ΑΝΘΡΩΠΟΥ-ΥΠΟΛΟΓΙΣΤΗ 2η ΕΚΔΟΣΗ*. Εταιρεία Αξιοποίησης και Διαχείρισης Περιουσίας Πανεπιστημίου Πατρών, 2016.

[30] I. SOMMERVILLE, *ΒΑΣΙΚΕΣ ΑΡΧΕΣ ΤΕΧΝΟΛΟΓΙΑΣ ΛΟΓΙΣΜΙΚΟΥ*. ΕΚΔΟΣΕΙΣ ΚΛΕΙΔΑΡΙΘΜΟΣ ΕΠΕ, 2011.

[31] S. Haykin, *Νευρωνικά Δίκτυα και Μηχανική Μάθηση*. Α. ΠΑΠΑΣΩΤΗΡΙΟΥ & ΣΙΑ Ι.Κ.Ε., 2010.

[32] A. Fox and D. A. Patterson, *ΤΕΧΝΟΛΟΓΙΑ ΑΝΑΠΤΥΞΗΣ ΛΟΓΙΣΜΙΚΟΥ ΩΣ ΥΠΗΡΕΣΙΑΣ: ΜΙΑ ΕΥΕΛΙΚΤΗ ΠΡΟΣΕΓΓΙΣΗ ΜΕ ΧΡΗΣΗ ΥΠΟΛΟΓΙΣΤΙΚΗΣ ΝΕΦΟΥΣ*. ΕΚΔΟΣΕΙΣ ΚΛΕΙΔΑΡΙΘΜΟΣ ΕΠΕ, 2017.