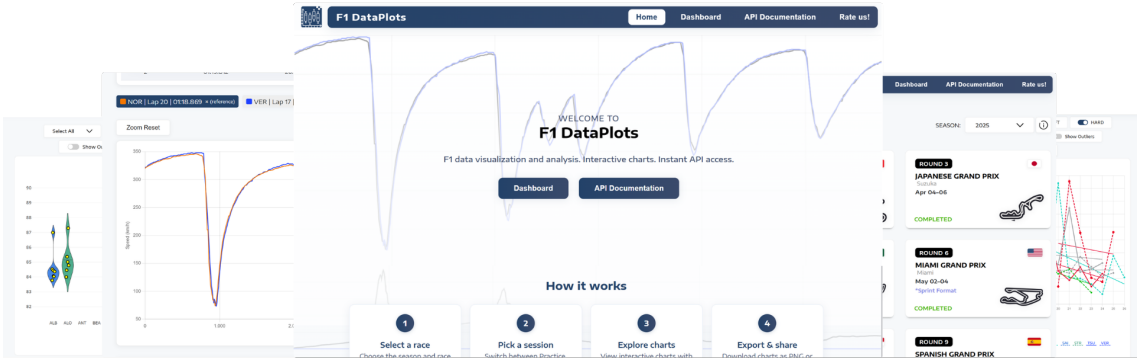
Διεθνές Πανεπιστήμιο Ελλάδος
Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

F1DataPlots: Εφαρμογή ιστού για την οπτικοποίηση δεδομένων ελεύθερων δοκιμών, κατατακτηρίων και αγώνων F1



Φοιτητής:
Κωνσταντίνος Βαϊράμης
Αριθμός Μητρώου: 2020018

Επιβλέπων:
Επίκουρος Καθηγητής
Στέφανος Ουγιάρογλου

Σεπτέμβριος 2025

F1DataPlots: Εφαρμογή ιστού για την οπτικοποίηση δεδομένων ελεύθερων δοκιμών, κατατακτηρίων και αγώνων F1

Κωδικός Δ.Ε.: 25135

Ονοματεπώνυμο φοιτητή: Κωνσταντίνος Βαϊράμης
Ονοματεπώνυμο εισηγητή: Στέφανος Ουγιάρογλου

Ημερομηνία ανάληψης: 27-02-2025
Ημερομηνία περάτωσης: 12-09-2025

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως διπλωματική εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

«Δεν απέτυχα. Απλώς βρήκα 10.000 τρόπους που δεν λειτουργούν.»
Τόμας Ά. Έντισον

Abstract

This diploma thesis focuses on the development of the web application “F1dataplots”, which leverages the FastF1 library to retrieve and process data from Formula 1 races, offering users a comprehensive environment for F1 data visualization and analysis through interactive charts. The user can visualize a wide range of data, from lap times to detailed telemetry elements of individual laps. The user interface has been designed with simplicity and usability in mind, making it accessible to different types of users, whether they are just fans of the sport or data analysts. The application provides organized access to a variety of features and enables the export of data and charts in different formats (PNG, CSV), facilitating further analysis or use in research and educational contexts. Additionally, the application offers a publicly available Web API, allowing anyone to access and utilize its data.

Περίληψη

Η παρούσα διπλωματική εργασία επικεντρώνεται στην ανάπτυξη της διαδικτυακής εφαρμογής «F1dataplots», η οποία αξιοποιεί τη βιβλιοθήκη FastF1 για την άντληση και επεξεργασία δεδομένων από αγώνες Formula 1 και προσφέρει στους χρήστες ένα ολοκληρωμένο περιβάλλον οπτικοποίησης και ανάλυσης δεδομένων Formula 1, μέσα από διαδραστικά γραφήματα. Ο χρήστης έχει τη δυνατότητα να οπτικοποίησει πληθώρα δεδομένων, από χρόνους γύρων έως και επιμέρους στοιχεία τηλεμετρίας γύρων. Η διεπαφή χρήστη έχει σχεδιαστεί με γνώμονα την απλότητα και την ευχρηστία, ώστε να είναι προσιτή σε διαφορετικά είδη χρηστών, είτε πρόκειται για απλούς φίλους του αθλήματος είτε για αναλυτές δεδομένων. Η εφαρμογή προσφέρει οργανωμένη πρόσβαση σε μια πληθώρα λειτουργιών και παρέχει τη δυνατότητα εξαγωγής δεδομένων και γραφημάτων σε διάφορες μορφές (PNG, CSV), διευκολύνοντας την περαιτέρω ανάλυση ή αξιοποίηση σε ερευνητικά και εκπαιδευτικά περιβάλλοντα. Η εφαρμογή διαθέτει ένα Web API, που είναι δημόσια διαθέσιμο και ο καθένας μπορεί να χρησιμοποιήσει τα δεδομένα της εφαρμογής.

Ευχαριστίες

Αρχικά, θα ήθελα να ευχαριστήσω την οικογένεια και τους φίλους μου για την αμέριστη υποστήριξη και τη συνεχή ενθάρρυνσή τους καθ' όλη τη διάρκεια των σπουδών μου. Τους ευχαριστώ για όλες τις στιγμές που περάσαμε μαζί, οι οποίες μου έδωσαν δύναμη και χαρά σε κάθε βήμα αυτής της διαδρομής.

Επιπλέον θα ήθελα να ευχαριστήσω την εταιρεία McLaren Applied και τον Oliver Marler (Product Manager – Training and Engineering Services – Motorsport) οι οποίοι με βοήθησαν να αποκτήσω μια πιο ολοκληρωμένη εικόνα για τα δεδομένα στη Formula 1 και για τις τεχνολογίες που αξιοποιούν.

Τέλος, θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα της διπλωματικής μου εργασίας, τον Επίκουρο Καθηγητή κ. Στέφανο Ουγιάρογλου, ο οποίος από την πρώτη στιγμή στάθηκε δίπλα μου, δείχνοντάς μου εμπιστοσύνη τόσο ως προς το θέμα της διπλωματικής μου, όσο και συνολικά στην πορεία ολοκλήρωσής της. Η καθοδήγηση, οι συμβουλές και η συνεχής υποστήριξή του αποτέλεσαν καθοριστικούς παράγοντες για την επιτυχή ολοκλήρωση της εργασίας μου.

Περιεχόμενα

Συντομογραφίες

API Application Programming Interface. 13, 14

ATLAS Advanced Telemetry Linked Acquisition System. 10, 11

CAN Controller Area Network. 9

DiL Driver-in-Loop. 11, 12

ECU Electronic Control Unit. 9

F1 Formula 1. vii, 1, 4, 9, 11

FIA Fédération Internationale de l'Automobile. 1--3

FP1 Free Practice 1. 6, 7, 13

FP2 Free Practice 2. 6, 13

FP3 Free Practice 3. 6, 13

RPM Revolutions Per Minute. 14

Γλωσσάρι

G-forces Οι δυνάμεις που δρουν σε έναν οδηγό ή αντικείμενο λόγω επιταχύνσεων και επιβραδύνσεων, μετρούμενες σε πολλαπλάσια της επιτάχυνσης της βαρύτητας ($g \approx 9.81\,m/s^2$). Στη Formula 1, οι οδηγοί συχνά υφίστανται υψηλές πλευρικές και διαμήκεις g-forces κατά τις στροφές, τα φρεναρίσματα και τις επιταχύνσεις.. 9

Grand Prix Ο όρος προέρχεται από τη Γαλλία (σημαίνει «μεγάλο βραβείο») και χρησιμοποιήθηκε πρώτη φορά σε αγώνα αυτοκινήτων το 1906. Στην Formula 1 χρησιμοποιείτε στην επίσημη ονομασία ενός αγώνα (π.χ. Dutch Grand Prix).. 2, 6, 7, 11

Pit Stop Σύντομη στάση (περίπου 2-3 δευτερόλεπτα) ενός μονοθεσίου στα pits κατά τη διάρκεια ενός αγώνα, για αλλαγή ελαστικών. 7, 13

Pit Wall Σημείο που κάθονται οι μηχανικοί αγώνα των οδηγών κατά τη διάρκεια του αγώνα. Από το pit wall η ομάδα παρακολουθεί τα δεδομένα τηλεμετρίας, επικοινωνεί με τον οδηγό και λαμβάνει στρατηγικές αποφάσεις.. 10

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

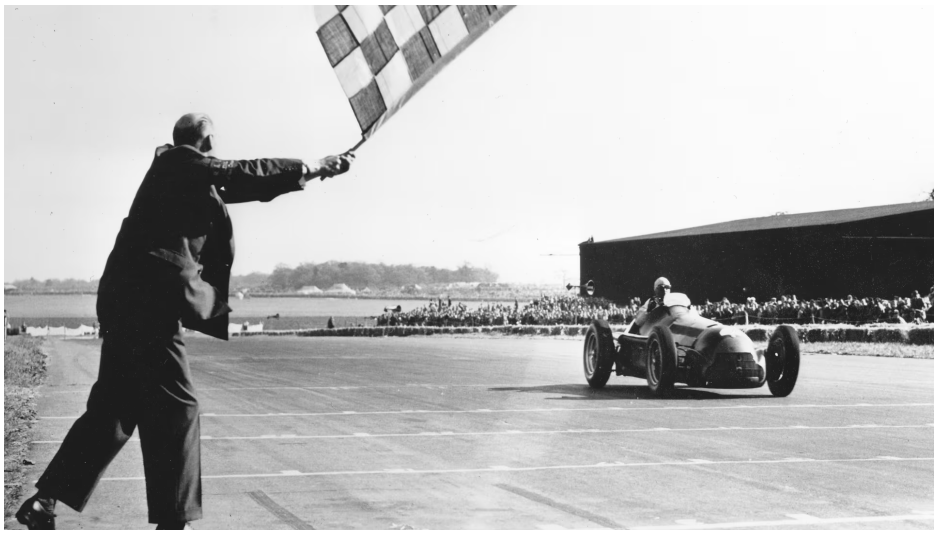
Κεφάλαιο 1

Εισαγωγή

1.1 Αγώνες Formula 1

Η F1 αποτελεί την κορωνίδα του μηχανοκίνητου αθλητισμού παγκοσμίως και θεωρείται ένα από τα πιο απαιτητικά και θεαματικά αθλήματα μηχανοκίνητου αθλητισμού. Διοργανώνεται υπό την αιγίδα της Διεθνούς Ομοσπονδίας Αυτοκινήτου (Fédération Internationale de l'Automobile (FIA)) και από την πρώτη της σεζόν, το 1950, έχει καταφέρει να ελκύσει εκατομμύρια θεατές σε όλο τον κόσμο. Οι αγώνες διεξάγονται σε διαφορετικές χώρες και πίστες, από ιστορικές διαδρομές όπως το Μόντε Κάρλο του Μονακό μέχρι σύγχρονες πίστες όπως αυτό του Άμπου Ντάμπι. Κάθε μια από αυτές προσφέρει το δικό της θέαμα και της δικές της προκλήσεις.

Ο πρώτος επίσημος αγώνας του παγκοσμίου πρωταθλήματος Formula 1 πραγματοποιήθηκε στις 13 Μαΐου 1950 στη ιστορική πίστα του Silverstone στη Μεγάλη Βρετανία. Εκείνη τη χρονιά συμμετείχαν κυρίως ευρωπαϊκές ομάδες, με την Alfa Romeo να κυριαρχεί και τον Giuseppe Farina να γίνεται ο πρώτος νικητής σε αγώνα F1. Από τότε μέχρι σήμερα, η Formula 1 εξελίχθηκε ραγδαία, τόσο σε αγωνιστικό όσο και σε τεχνολογικό επίπεδο, καταφέρνοντας να έχει αποκτήσει εκατομμύρια υποστηρικτές.



Σχήμα 1.1: Ο πρώτος επίσημος αγώνας του Παγκοσμίου Πρωταθλήματος Formula 1 το 1950 στο Silverstone (Πηγή: Formula 1)

1.1.1 Δομή της Formula 1

Στην σημερινή εποχή κάθε σεζόν της Formula 1 περιλαμβάνει 24 αγώνες (Grand Prix), οι οποίοι διεξάγονται σε διάφορες πίστες ανά τον κόσμο. Στο παρελθόν ο αριθμός των αγώνων ήταν μικρότερος, ωστόσο τα τελευταία χρόνια το πρωτάθλημα έχει επεκταθεί σημαντικά, περιλαμβάνοντας καινούριους προορισμούς σε διαφορετικές ηπείρους. Κάθε ομάδα της Formula 1 έχει στο δυναμικό της δύο βασικούς οδηγούς, με τους οποίους συμμετέχει σε κάθε αγώνα. Επιπλέον, οι ομάδες διαθέτουν και έναν τρίτο οδηγό, τον αποκαλούμενο «test driver», ο οποίος συμβάλλει σημαντικά στην εξέλιξη και τη βελτίωση του μονοθεσίου τόσο κατά τις χειμερινές δοκιμές πριν την έναρξη της σεζόν, όσο και κατά τη διάρκεια των αγώνων.

Η βαθμολογία για τους οδηγούς διαμορφώνεται με βάση τη θέση τερματισμού τους σε κάθε αγώνα, σύμφωνα με το σύστημα βαθμολόγησης που έχει θεσπίσει η FIA. Στην αγωνιστική σεζόν του 2025, συμμετείχαν 10 ομάδες με 20 συνολικά οδηγούς, ωστόσο από το 2026 στην Formula 1 θα ενταχθεί και 11η ομάδα και στόχος της FIA είναι να αυξηθεί ο αριθμός τους. Μέσα στη Formula 1 διεξάγονται δύο ξεχωριστά πρωταθλήματα: το Παγκόσμιο Πρωτάθλημα Οδηγών και το Παγκόσμιο Πρωτάθλημα Κατασκευαστών (Ομάδων). Κάθε οδηγός συλλέγει βαθμούς ανάλογα με τη θέση στην οποία τερματίζει σε κάθε αγώνα, ενώ ταυτόχρονα οι βαθμοί που συγκεντρώνουν οι δύο οδηγοί κάθε ομάδας προσμετρούνται στη συνολική βαθμολογία της ομάδας. Στο τέλος της σεζόν, ο οδηγός με τους περισσότερους βαθμούς ανακηρύσσεται Παγκόσμιος Πρωταθλητής Οδηγών, ενώ η ομάδα με τους περισσότερους βαθμούς κατακτά το Πρωτάθλημα Κατασκευαστών.

Η σεζόν ξεκινά συνήθως στις αρχές της άνοιξης και ολοκληρώνεται στις αρχές του χειμώνα. Οι αγώνες δεν διεξάγονται σε 24 συνεχόμενα Σαββατοκύριακα, αλλά κατανέμονται σε όλη τη διάρκεια της σεζόν, από την άνοιξη τον χειμώνα. Αυτό γίνεται ώστε να υπάρχει χρόνος ξεκούρασης για τους οδηγούς και τις ομάδες αλλά και για να διευκολυνθεί η διαδικασία των logistics, καθώς η μεταφορά του εξοπλισμού από χώρα σε χώρα (για παράδειγμα, από την Κίνα στην Ισπανία) μέσα

σε λίγες ημέρες είναι ιδιαίτερα απαιτητική.

Η Formula 1 λειτουργεί υπό ένα αυστηρό πλαίσιο κανονισμών που θεσπίζει η Διεθνής Ομοσπονδία Αυτοκινήτου (FIA). Οι κανονισμοί αυτοί αφορούν τόσο την ασφάλεια των οδηγών και των θεατών, όσο και τα τεχνικά χαρακτηριστικά των μονοθεσίων, τα οικονομικά των ομάδων αλλά και τη γενικότερη διεξαγωγή των αγώνων. Στόχος τους είναι να εξασφαλίζεται δίκαιος ανταγωνισμός, να ενθαρρύνεται η καινοτομία και να διατηρείται το άθλημα προσιτό και ελκυστικό για όλες τις ομάδες που συμμετέχουν και σε αυτές που θέλουν να εισέλθουν μελλοντικά.

1.1.2 Ιστορικά στοιχεία της Formula 1

Οδηγοί

Η ιστορία της Formula 1 είναι γεμάτη από εμβληματικές προσωπικότητες και ιστορικές στιγμές που καθόρισαν το άθλημα. Ονόματα όπως ο Juan Manuel Fangio, ο Ayrton Senna, ο Michael Schumacher και ο Lewis Hamilton έχουν αφήσει ανεξίτηλο το αποτύπωμά τους.

Ο πρώτος Παγκόσμιος Πρωταθλητής της Formula 1 ήταν ο Ιταλός Giuseppe “Nino” Farina, ο οποίος κατέκτησε τον τίτλο το 1950 οδηγώντας για λογαριασμό της Alfa Romeo.

Τα επόμενα χρόνια, η σκυτάλη πέρασε στον Αργεντινό Juan Manuel Fangio, ο οποίος θεωρείται από πολλούς ως ο μεγαλύτερος οδηγός όλων των εποχών. Μέσα σε μόλις επτά σεζόν, κατέκτησε πέντε παγκόσμια πρωταθλήματα — με τέσσερις διαφορετικές ομάδες — και πέτυχε εκπληκτικά ρεκόρ για την τότε εποχή.

Στις δεκαετίες που ακολούθησαν, νέα ονόματα πρωταγωνίστησαν στο άθλημα όπως ο Jackie Stewart, ο Niki Lauda και ο Alain Prost που έθεσαν νέα πρότυπα, είτε με την ταχύτητα και τη συνέπειά τους είτε με την αποφασιστικότητα και το πάθος τους.

Ο Σκωτσέζος Jackie Stewart στέφθηκε τρεις φορές Παγκόσμιος Πρωταθλητής (1969, 1971, 1973) και εκτός από τις αγωνιστικές του επιτυχίες, έμεινε στην ιστορία ως ο άνθρωπος που έκανε την ασφάλεια κεντρικό ζήτημα στη Formula 1. Με την επιμονή του για καλύτερες πίστες, βελτιωμένα μέτρα προστασίας και αυστηρότερους κανονισμούς, βοήθησε να σωθούν αμέτρητες ζωές και καθόρισε την εξέλιξη του αθλήματος.

Ο Niki Lauda, πρωταθλητής το 1975, 1977 και 1984, έγραψε μια από τις πιο συγκλονιστικές σελίδες στην ιστορία του αθλήματος. Μετά από ενώ σοβαρό και σχεδόν θανατηφόρο ατύχημα το 1976, επέστρεψε πιο δυνατός από ποτέ και ανακηρύχτηκε ξανά πρωταθλητής.

Σχήμα 1.2: Ιστορικοί οδηγοί της Formula 1. Από αριστερά προς τα δεξιά: Giuseppe Farina, Juan Manuel Fangio, Jackie Stewart και Niki Lauda (Πηγή: Formula 1)

Στη δεκαετία του 1980, το προσκήνιο κατέλαβε ο Alain Prost, ο οποίος κατέκτησε τέσσερις τίτλους (1985, 1986, 1989, 1993). Γνωστός ως «The Professor», ο Γάλλος οδηγός είχε μια διαφορετική προσέγγιση στους αγώνες, με υπολογισμένη στρατηγική και εκπληκτική διαχείριση ρυθμού αγώνα. Ο Prost υπήρξε υπόδειγμα συνέπειας και αποτελεσματικότητας, ενώ οι μάχες του με τον Ayrton Senna στη McLaren έμειναν αξέχαστες και καθόρισαν μια ολόκληρη εποχή της Formula 1.

Ιδιαίτερη μνεία αξίζει στον Ayrton Senna, έναν από τους πιο χαρισματικούς και αγαπητούς οδηγούς όλων των εποχών. Ο Βραζιλιάνος κατέκτησε 3 παγκόσμιους τίτλους (1988, 1990, 1991) με τη McLaren και έμεινε στην ιστορία για την ταχύτητά του, κυρίως στις κατατακτήριες δοκιμές, αλλά και για τη μαχητικότητά του μέσα στην πίστα. Ο πρόωρος θάνατός του στο Grand Prix του Σαν Μαρίνο το 1994 συγκλόνισε τον κόσμο του μηχανοκίνητου αθλητισμού και αποτέλεσε ορόσημο για τη βελτίωση των κανονισμών ασφαλείας στη Formula 1.

Λίγα χρόνια αργότερα ο κόσμος τις F1 ανήκε στον εντυπωσιακό Michael Schumacher, ο οποίος κατέκτησε επτά παγκόσμιους τίτλους (δύο με την Benetton και πέντε συνεχόμενους με τη Ferrari). Ο Γερμανός οδηγός κατέρριψε σωρεία ρεκόρ και καθιερώθηκε ως ο πιο επιτυχημένος οδηγός της εποχής του.

Μετά το φαινόμενο Schumacher, τη σκηνή κατέλαβε ο Lewis Hamilton, που και αυτός έγραψε ιστορία με επτά παγκοσμίους τίτλους (2008, 2014, 2015, 2017–2020). Ο Βρετανός κατέκτησε το πρώτο του πρωτάθλημα με τον πιο δραματικό τρόπο, στην τελευταία στροφή του τελευταίου αγώνα της σεζόν, ενώ το 2021 έχασε τον τίτλο με εξίσου δραματικό τρόπο στον τελευταίο γύρο του Grand Prix του Άμπου Ντάμπι. Ο Hamilton συνδύασε την αγωνιστική επιτυχία με κοινωνική επιρροή, αποτελώντας πρότυπο όχι μόνο για την ταχύτητα και τη συνέπειά του, αλλά και για τον ακτιβισμό του εντός και εκτός πίστας. Είναι ακόμη και σήμερα ο οδηγός που έχει καταφέρει να σπάσει σχεδόν όλα τα ρεκόρ του Schumacher και με την αγωνιστική του δράση να συνεχίζεται, δεν αποκλείεται στο μέλλον να καταρρίψει και το φράγμα των οκτώ παγκόσμιων τίτλων.

Τέλος, η νέα γενιά έχει δείξει ότι το μέλλον της Formula 1 ανήκει σε νέους αστέρες. Ο Max Verstappen έχει ήδη σπάσει πολλά ρεκόρ και σε νεαρή ηλικία έχει καταφέρει να καθιερωθεί ανάμεσα στους κορυφαίους όλων των εποχών. Με τρεις τίτλους παγκόσμιου πρωταθλητή, περισσότερες από 60 νίκες και σχεδόν 110 παρουσίες στο βάθρο μετά από περίπου 200 αγωνιστικά Σαββατοκύριακα, ο Ολλανδός οδηγός έχει ήδη μπει στο Πάνθεον των καλύτερων οδηγών. [1]

Σχήμα 1.3: Ιστορικοί οδηγοί της Formula 1. Από αριστερά προς τα δεξιά: Alain Prost, Ayrton Senna, Michael Schumacher, Lewis Hamilton και Max Verstappen (Πηγή: Formula 1)

Ομάδες

Η Formula 1 δεν θα ήταν το ίδιο χωρίς τις εμβληματικές ομάδες που διαμόρφωσαν την ιστορία της και συνεχίζουν να γράφουν το μέλλον του αθλήματος. Ανάμεσα σε αυτές ξεχωρίζουν η Ferrari, η McLaren, η Mercedes, η Red Bull και η Williams, καθεμία με τη δική της ξεχωριστή ιστορία.

Η Scuderia Ferrari είναι η μοναδική ομάδα που έχει συμμετάσχει σε κάθε σεζόν από το 1950 έως σήμερα. Με την εμβληματική κόκκινη εμφάνιση, η Ferrari έχει κατακτήσει πολυάριθμους τίτλους και είχε στο δυναμικό της θρύλους οδηγούς όπως ο Alberto Ascari, ο Niki Lauda και ο Michael Schumacher. Το τελευταίο της Παγκόσμιο Πρωτάθλημα Οδηγών ήρθε με τον Kimi Räikkönen το 2007, ενώ το τελευταίο Πρωτάθλημα Κατασκευαστών το 2008. Παρά τα σκαμπανεβάσματα, το όνομα Ferrari παραμένει άρρηκτα συνδεδεμένο με τη Formula 1.

Η McLaren, ιδρυμένη το 1966 από τον Νεοζηλανδό Bruce McLaren, είναι η δεύτερη παλαιότερη ομάδα στο grid. Η πρώτη της νίκη ήρθε το 1968 στο Βελγικό Grand Prix και από τότε έχει σημειώσει περισσότερες από 200 νίκες, 9 Πρωταθλήματα Κατασκευαστών και 12 Πρωταθλήματα Οδηγών. Στη McLaren αγωνίστηκαν μερικοί από τους σπουδαιότερους οδηγούς της ιστορίας, όπως οι Ayrton Senna, Alain Prost και Lewis Hamilton, χαρίζοντας στιγμές δόξας στους φίλους του αθλήματος.

Η Mercedes εμφανίστηκε αρχικά το 1954, αλλά επέστρεψε δυναμικά μόλις το 2010, μετά την εξαγορά της ομάδας Brawn GP. Στην υβριδική εποχή της Formula 1 (2014–2021) κυριάρχησε πλήρως, κατακτώντας 7 Πρωταθλήματα Οδηγών – έξι με τον Lewis Hamilton και ένα με τον Nico Rosberg – καθώς και 8 συνεχόμενα Πρωταθλήματα Κατασκευαστών. Τα «Ασημή βέλη» (Silver Arrows) έγραψαν έτσι μια ολόκληρη εποχή κυριαρχίας.

Η Red Bull Racing αποτελεί τη νεότερη από τις κορυφαίες ομάδες, με την πορεία της να ξεκινά το 2005. Μέσα σε σύντομο χρονικό διάστημα, η Red Bull αναδείχθηκε σε υπερδύναμη, κατακτώντας 6 Πρωταθλήματα Κατασκευαστών και 8 Πρωταθλήματα Οδηγών, χάρη στους Sebastian Vettel και Max Verstappen.

Τέλος, η Williams ιδρύθηκε το 1978 από τους Frank Williams και Patrick Head. Με 9 Πρωταθλήματα Κατασκευαστών και 7 Πρωταθλήματα Οδηγών, η Williams γνώρισε τις χρυσές της στιγμές στις δεκαετίες του ’80 και ’90 με οδηγούς όπως ο Nelson Piquet, ο Nigel Mansell και ο Damon Hill. Αν και τα τελευταία χρόνια δεν βρίσκεται στις πρώτες θέσεις, εξακολουθεί να αποτελεί ένα από τα ιστορικότερα ονόματα του αθλήματος.[2]

1.1.3 Δομή ενός αγωνιστικού τριήμερου (Παρασκευή - Σάββατο - Κυριακή)

Κάθε Grand Prix διαρκεί τρεις ημέρες και περιλαμβάνει συγκεκριμένες διαδικασίες που επιτρέπουν στις ομάδες να προετοιμαστούν για των αγώνα. [3]

Ελεύθερες δοκιμές (Free Practice – FP1, FP2, FP3)

Οι ελεύθερες δοκιμές πραγματοποιούνται Παρασκευή και Σάββατο και έχουν ως στόχο τη συλλογή δεδομένων και τη ρύθμιση των μονοθεσίων εν όψη των κατατακτήριων και του αγώνα.

- **FP1:** Η πρώτη 60λεπτη διαδικασία της Παρασκευής, χρησιμοποιείται από τις ομάδες ώστε να δοκιμάσουν τις αρχικές ρυθμίσεις, να αξιολογούν την απόδοση των ελαστικών και να συλλέξουν δεδομένα για την πίστα.
- **FP2:** Πραγματοποιείται επίσης την Παρασκευή, το απόγευμα. Σε αυτό το στάδιο οι ομάδες κάνουν πιο εκτεταμένες δοκιμές στρατηγικών και προσομοιώσεις αγώνα καθώς η ώρα που διεξάγεται είναι κοντά στην ώρα του αγώνα, συνεπώς οι συνθήκες είναι παρόμοιες με τον αγώνα.
- **FP3:** Η τελευταία ελεύθερη διαδικασία είναι το Σάββατο το πρωί. Αποτελεί την τελευταία ευκαιρία για τις ομάδες και τους οδηγούς ώστε να ρυθμίσουν τα μονοθέσια τους πριν τις κατατακτήριες. Συνήθως επικεντρώνεται σε γρήγορους γύρους προσομοίωσης κατατακτήριων.

Κατατακτήριες (Qualifying – Q1, Q2, Q3)

Οι κατατακτήριες πραγματοποιούνται το Σάββατο και καθορίζουν τη σειρά εκκίνησης του αγώνα. Το format αποτελείται από τρία σκέλη με σταδιακό αποκλεισμό:

- **Q1:** Διάρκεια 18 λεπτών. Όλοι οι οδηγοί προσπαθούν να πετύχουν γρήγορο γύρο. Οι πέντε πιο αργοί αποκλείονται και κατατάσσονται στις τελευταίες θέσεις.
- **Q2:** Διάρκεια 15 λεπτών. Οι υπόλοιποι οδηγοί συνεχίζουν την προσπάθεια. Οι πέντε πιο αργοί αποκλείονται, καθορίζοντας τις θέσεις 11-15.
- **Q3:** Διάρκεια 12 λεπτών. Οι δέκα ταχύτεροι οδηγοί διεκδικούν την pole position, δηλαδή την πρώτη θέση εκκίνησης του αγώνα και κατατάσσονται ανάλογα με τον χρόνο που έχουν πετύχει.

Αγώνας (Race)

Ο αγώνας πραγματοποιείται την Κυριακή και είναι το αποκορύφωμα του τριημέρου. Η απόσταση ενός Grand Prix είναι περίπου στα 305 χιλιόμετρα ή 2 ώρες μέγιστης διάρκειας (σε περίπτωση που υπάρχουν συμβάντα και διακοπή του αγώνα). Οι οδηγοί ξεκινούν με βάση τη σειρά που πήραν στις κατατακτήριες. Κατά τη διάρκεια του αγώνα εφαρμόζονται διαφορετικές στρατηγικές ελαστικών και Pit Stop, ενώ οι καιρικές συνθήκες και τα συμβάντα στην πίστα (όπως αυτοκίνητα ασφαλείας) μπορούν να αλλάξουν την έκβαση του αποτελέσματος. Οι βαθμοί απονέμονται στους δέκα πρώτους οδηγούς με σύστημα 25-18-15-12-10-8-6-4-2-1. [3]

Αγώνας Sprint (Sprint Race)

Από το 2022, ορισμένα αγωνιστικά τριήμερα στη Formula 1 περιλαμβάνουν έναν αγώνα Sprint, διαφοροποιώντας τη τυπική δομή του Σαββατοκύριακου. Το Sprint είναι ένας σύντομος αγώνας, διάρκειας 100 χιλιομέτρων (ή περίπου 30 λεπτών), που πραγματοποιείται το Σάββατο και προσφέρει βαθμούς στους οκτώ πρώτους οδηγούς. Η εισαγωγή αυτού το μικρού αγώνα έγινε για να προσθέσει περισσότερη δράση και ενδιαφέρον για τους θεατές του αθλήματος. [3]

Η σειρά εκκίνησης του Sprint καθορίζεται από μια ξεχωριστή διαδικασία κατατακτήριων (Sprint Shootout), ενώ η σειρά εκκίνησης του κυρίως αγώνα την Κυριακή καθορίζεται από την κανονική διαδικασία κατατακτήριων.

Έτσι σε αγωνιστικά τριήμερα που περιλαμβάνουν αγώνα Sprint, το πρόγραμμα διαμορφώνεται ως εξής:

- **FP1:** Η πρώτη και μοναδική περίοδος ελεύθερων δοκιμών πραγματοποιείται την Παρασκευή, όπου οι ομάδες και οι οδηγοί προετοιμάζουν τα μονοθέσια για το υπόλοιπο του τριημέρου.
- **Sprint Qualifying (Sprint Shootout):** Πραγματοποιείται την Παρασκευή το απόγευμα και καθορίζει τη σειρά εκκίνησης για το Sprint.
  - **SQ1:** Διάρκεια 12 λεπτών. Συμμετέχουν όλοι οι οδηγοί και οι πέντε πιο αργοί αποκλείονται.
  - **SQ2:** Διάρκεια 10 λεπτών. Συνεχίζουν οι υπόλοιποι και στο τέλος αποκλείονται οι πέντε πιο αργοί.

  - **SQ3:** Διάρκεια 8 λεπτών. Οι δέκα ταχύτεροι οδηγοί διεκδικούν την pole position για το Sprint.

- **Sprint:** Ένας σύντομος αγώνας περίπου 100 χιλιομέτρων που διεξάγεται το Σάββατο το πρωί, με βαθμούς για τους οκτώ πρώτους και χωρίς υποχρεωτικό pit stop.
- **Qualifying:** Η κλασική διαδικασία κατατακτήριων που πραγματοποιείται κανονικά το Σάββατο και ορίζει τη σειρά εκκίνησης του κυρίως αγώνα της Κυριακής.
- **Race:** Ο κύριος αγώνας της Κυριακής.

1.1.4 Ο ρόλος της Formula 1 στην τεχνολογική πρόοδο και την κοινωνία

Η F1 δεν αποτελεί μόνο άθλημα ταχύτητας αλλά ένα κέντρο τεχνολογικής εξέλιξης και καινοτομίας. Τα μονοθέσια σχεδιάζονται και κατασκευάζονται με βάση τις πιο σύγχρονες προδιαγραφές στη μηχανολογία και την αεροδυναμική, ενώ οι ομάδες επενδύουν τεράστιους πόρους στην ανάπτυξη λύσεων που θα τους δώσουν το ανταγωνιστικό πλεονέκτημα. Πολλές από τις τεχνολογίες που αναπτύχθηκαν για τη Formula 1 βρήκαν μετέπειτα εφαρμογές στα αυτοκίνητα παραγωγής, καθιστώντας το άθλημα ένα πεδίο έρευνας και ανάπτυξης με άμεσο αντίκτυπο στην αυτοκινητοβιομηχανία και συνεπώς στην καθημερινότητα του ανθρώπου. Ένα χαρακτηριστικό παράδειγμα είναι το σύστημα KERS (Kinetic Energy Recovery System). Εισήχθη στην F1 το 2009 και επιτρέπει την ανάκτηση κινητικής ενέργειας κατά την πέδηση, η οποία αποθηκεύεται σε μπαταρία και χρησιμοποιείται για επιπλέον ισχύς. Αργότερα, τεχνολογίες βασισμένες στο KERS υιοθετήθηκαν από αυτοκινητοβιομηχανίες. Για παράδειγμα, αρκετά h hybrid ή plug in hybrid επιβατικά οχήματα (όπως το Volvo XC90) χρησιμοποιούν συστήματα ανάκτηση ενέργειας κατά το φρενάρισμα με σκοπό τη βελτίωση της οικονομίας καυσίμου και της απόδοσης.

Η Formula 1 έχει επίσης τεράστιο οικονομικό και κοινωνικό αντίκτυπο. Κάθε αγώνας δημιουργεί σημαντικά έσοδα για τις διοργανώτριες πόλεις, ενισχύει τον τουρισμό και προβάλλει τις χώρες που φιλοξενούν τα Grand Prix σε παγκόσμιο επίπεδο. Παράλληλα, το άθλημα συνδέει ανθρώπους από διαφορετικές κουλτούρες και κοινωνικές τάξεις, δημιουργώντας μια παγκόσμια κοινότητα φιλάθλων που παρακολουθούν με πάθος τους αγώνες.

1.2 Δεδομένα Αγώνων F1

Η λειτουργία ενός σύγχρονου μονοθεσίου Formula 1 δεν είναι απλή υπόθεση. Πρόκειται για τις πιο σύνθετες και περίπλοκες μηχανές αυτοκινήτων στον κόσμο και απαιτεί όχι μόνο τις δεξιότητες του οδηγού, αλλά και τεράστιο όγκο δεδομένων για να λειτουργήσει σωστά. Στη σημερινή εποχή, τα δεδομένα είναι εξίσου κρίσιμα και σημαντικά με την ίδια την ταχύτητα που χρειάζονται τα μονοθέσια.

Οι ομάδες έχουν πρόσβαση σε τεράστιο όγκο δεδομένων σε πραγματικό χρόνο, που τους επιτρέπουν να παρακολουθούν κάθε μέρος των μονοθεσίων τους κατά τη διάρκεια των αγωνιστικών τριημέρων.

Αυτός ο τεράστιος όγκος πληροφορίας δεν είναι απλά μια στατιστική ανάλυση, παρέχει στους μηχανικούς και στους οδηγούς τα εργαλεία για να λαμβάνουν αποφάσεις σε γρήγορο χρόνο και να μεγιστοποιούν την απόδοση με στόχο τις νίκες. Στην σημερινή εποχή όπου ο χρόνος στην πίστα είναι περιορισμένος και η τεχνολογία εξελίσσεται με ταχύτατους ρυθμούς, τα δεδομένα έχουν γίνει αναπόσπαστο κομμάτι της επιτυχίας. Χωρίς αξιόπιστα δεδομένα, ένας οδηγός και μια ομάδα F1 δεν μπορούν να αποδώσουν στο μέγιστο των δυνατοτήτων τους.

1.2.1 Πηγές και Τύποι Δεδομένων

Τα δεδομένα παράγονται κυρίως από τους αισθητήρες που είναι εγκατεστημένοι πάνω στα μονοθέσια. Ένα σύγχρονο μονοθέσιο F1 μπορεί να έχει πάνω από 250 διαφορετικούς αισθητήρες (ένα τυπικό αυτοκίνητο έχει περίπου 30, ενώ ένα πολυτελές αυτοκίνητο μπορεί να έχει από 100 έως 200 αισθητήρες) που καταγράφουν θερμοκρασίες, πιέσεις, δυνάμεις επιτάχυνσης (G-forces), κατάσταση ελαστικών, λειτουργία του κινητήρα και των υβριδικών μονάδων καθώς και την αεροδυναμική απόδοση [4].

Οι αισθητήρες αυτοί χωρίζονται σε τρεις βασικές κατηγορίες:

- **Control sensors:** Ρυθμίζουν και ελέγχουν συστήματα του μονοθεσίου.
- **Instrumentation sensors:** Παρέχουν πληροφορίες για την ανάλυση δεδομένων και τις ρυθμίσεις.
- **Monitoring sensors:** Παρακολουθούν την κατάσταση κρίσιμων εξαρτημάτων για λόγους ασφάλειας και αξιοπιστίας.

Όλα αυτά συνδέονται με την Μονάδα Ηλεκτρονικού Ελέγχου (Electronic Control Unit (ECU)) μέσω αναλογικών συνδέσεων ή δικτύων CAN Bus (Controller Area Network (CAN)). Σε ένα τυπικό μονοθέσιο λειτουργούν έως και 17 διαφορετικά CAN bus, που επικοινωνούν ταυτόχρονα με πολλαπλές μονάδες ελέγχου.

Σχήμα 1.4: Η Μονάδα Ηλεκτρονικού Ελέγχου TAG-320B από την εταιρεία McLaren Applied (Πηγή: McLaren Applied)

1.2.2 Όγκος Δεδομένων

Ένα μονοθέσιο μπορεί να παράγει πάνω από 1 τεραμπάιτ δεδομένων ανά αγωνιστικό τριήμερο (συμπεριλαμβανομένων βίντεο και άλλων πληροφοριών). Κατά τη διάρκεια ενός γύρου, η «ζωντανή» ροή δεδομένων τηλεμετρίας φτάνει κατά μέσο όρο τα 30 MB, ενώ όταν το μονοθέσιο επιστρέφει στα pits και γίνεται μεταφόρτωση μέσω της φυσικής σύνδεσης, το μέγεθος αυτό πολλαπλασιάζεται. Σε αγώνες όπως το Μεξικό, η συνολική μεταφορά δεδομένων μεταξύ πίστας και εργοστασίων της Mercedes σε Brackley και Brixworth έφτασε τα 11 τεραμπάιτ μέσα σε ένα Σαββατοκύριακο.

1.2.3 Μετάδοση και Ανάλυση των δεδομένων

Η τηλεμετρία αποστέλλεται σε πραγματικό χρόνο στο Pit Wall αλλά και στα εργοστάσια των ομάδων. Σε ευρωπαϊκούς αγώνες, η καθυστέρηση μετάδοσης είναι της τάξης των 10 ms, ουσιαστικά στιγμιαία. Ακόμη και σε αγώνες μακρινών αποστάσεων όπως η Ιαπωνία ή η Αυστραλία, η καθυστέρηση σπάνια ξεπερνά τα 300 ms[4] [5].

Το Advanced Telemetry Linked Acquisition System (ATLAS) είναι ένα προηγμένο λογισμικό που έχει σχεδιαστεί για τη συλλογή, διανομή, οπτικοποίηση και ανάλυση δεδομένων από συστήματα ελέγχου, όπως αυτά που χρησιμοποιούνται στον μηχανοκίνητο αθλητισμό. Είναι ένα ισχυρό εργαλείο ανάλυσης τηλεμετρίας, που επιτρέπει στους αναλυτές δεδομένων και τις ομάδες μηχανικών να παρακολουθούν σε πραγματικό χρόνο κρίσιμες παραμέτρους των μονοθέσιων, να συγκρίνουν δεδομένα γύρων, να εντοπίζουν συσχετισμούς και να προβλέπουν επιδόσεις. Με υψηλό βαθμό παραμετροποίησης, το ATLAS προσφέρει πολλαπλές οπτικοποίησης και δυνατότητα εκτέλεσης μοντέλων MATLAB/Simulink πάνω σε πραγματικά δεδομένα. Το συγκεκριμένο λογισμικό έχει δημιουργηθεί από την εταιρεία McLaren Applied και χρησιμοποιείται ευρέως για την ανάλυση των δεδομένων τόσο στην πίστα όσο και στο εργοστάσιο από της ομάδες της Formula 1 [6].

Πρέπει να σημειωθεί ότι το ATLAS δεν είναι ένα ελεύθερα διαθέσιμο λογισμικό. Ωστόσο, κατόπιν επικοινωνίας με την εταιρεία McLaren Applied, παραχωρήθηκε προσωρινή εκπαιδευτική άδεια χρήσης για τις ανάγκες της παρούσας διπλωματικής εργασίας. Η άδεια αυτή επέτρεψε την αξιοποίηση του λογισμικού, διευκολύνοντας την κατανόηση των εργαλείων και των δεδομένων που χρησιμοποιούνται στην Formula 1.

1.2.4 Προσομοιωτές Οδηγών και Εικονικές Προσομοιώσεις

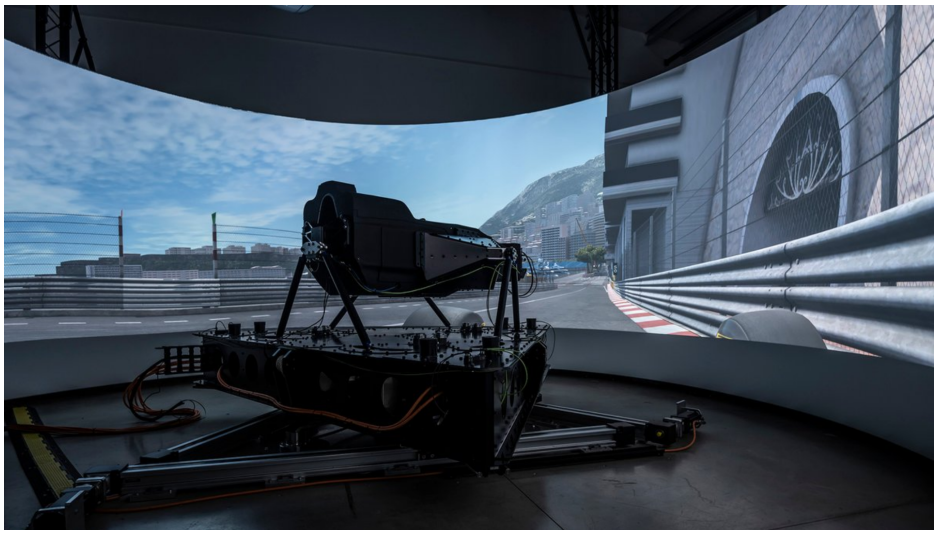
Driver-in-Loop (DiL) Προσομοιωτές

Ο προσομοιωτής DiL είναι ουσιαστικά μια εικονική πίστα δοκιμών που αναπαράγει με κάθε λεπτομέρεια το μονοθέσιο και την πίστα. Η εγκατάσταση μοιάζει σε φιλοσοφία με έναν επαγγελματικό προσομοιωτή αεροπλάνων, αλλά με cockpit ενός μονοθεσίου F1[7].

- Οι πίστες δημιουργούνται με LiDAR scanning, που παράγει τρισδιάστατους χάρτες με υψηλή ακρίβεια, καταγράφοντας κάθε καμπύλη, κάθε kerb και κάθε αλλαγή υψομέτρου.
- Το περιβάλλον του προσομοιωτή δημιουργείται με ίδιο σασί, τιμόνι και πεντάλ όπως το πραγματικό μονοθέσιο. Οι οδηγοί συχνά τρέχουν με πλήρη αγωνιστική εξάρτυση για να συγκεντρωθούν ακόμα περισσότερο.
- Σε μία τυπική συνεδρία, μπορούν να ολοκληρωθούν περισσότερα χιλιόμετρα από όσο διαρκεί ένας ολόκληρος αγώνας.

Η ακρίβεια του προσομοιωτή εξαρτάται από το πόσο καλά έχει γίνει το «correlation» με το πραγματικό αυτοκίνητο. Η ομάδα επενδύει σημαντικό χρόνο για να διασφαλίσει ότι οι αλλαγές σε ρυθμίσεις έχουν το ίδιο αποτέλεσμα στο DiL όπως και στην πραγματική πίστα [7].

Οι DiL συνεδρίες παίζουν κρίσιμο ρόλο ειδικά σε νέες πίστες χωρίς ιστορικά δεδομένα. Για ένα Grand Prix, το πρόγραμμα μπορεί να περιλαμβάνει 2 ημέρες προσομοίωσης DiL που αντιστοιχούν σε περίπου 450 εικονικούς γύρους, με τους εφεδρικούς οδηγούς να κάνουν τον κύριο όγκο, ενώ οι αγωνιστικοί οδηγοί αφιερώνουν χρόνο για να μάθουν τη χάραξη και να δουλέψουν σε ρυθμίσεις.



Σχήμα 1.5: Η Μονάδα Ηλεκτρονικού Ελέγχου TAG-320B από την εταιρεία McLaren Applied (Πηγή: Autosport)

Υπολογιστικές Προσομοιώσεις

Παράλληλα με τον προσομοιωτή οδηγού (DiL), οι ομάδες χρησιμοποιούν υπολογιστικές προσομοιώσεις που εκτελούνται πλήρως στον υπολογιστή χωρίς οδηγό (offline simulations) [7].

- Μπορούν να τρέξουν εκατοντάδες χιλιάδες εικονικούς γύρους βασισμένους σε δεδομένα από το DiL πριν από ένα Grand Prix, παράγοντας τεραμπάιτ δεδομένων.
- Οι εικονικοί γύροι μπορούν να εκτελούνται αρκετά γρηγορά και σε παράλληλες διεργασίες, δίνοντας σε λίγες ώρες όγκο πληροφορίας που θα χρειαζόταν εβδομάδες στην πραγματική πίστα.
- Οι μηχανικοί των μονοθεσίων εξετάζουν μικρές αλλαγές σε ρυθμίσεις και συγκρίνουν τα αποτελέσματα σε γραφήματα που συχνά επικαλύπτονται με δεδομένα από τον προσομοιωτή οδηγού και το πραγματικό μονοθέσιο για πλήρη ανάλυση.

Ενσωμάτωση Δεδομένων

Το πιο σημαντικό στοιχείο είναι ότι όλα τα δεδομένα από τις προσομοιώσεις (DiL και υπολογιστικές) συνδυάζονται με την πραγματική τηλεμετρία. Οι προσομοιώσεις δεν αντικαθιστούν την πίστα, αλλά κατευθύνουν την ομάδα στην επιλογή σωστής κατεύθυνσης και τους επιτρέπουν να φτάσουν την Παρασκευή πριν τον αγώνα, με ρυθμίσεις και στρατηγική που έχουν δοκιμαστεί νωρίτερα και βρίσκονται σε σωστή κατεύθυνση.

Κατά την διάρκεια ενός αγωνιστικού τριήμερου οι εφεδρικοί οδηγοί των ομάδων βρίσκονται στο εργοστάσιο δοκιμάζοντας νέες προσομοιώσεις για να βοηθήσουν την ομάδα και τους οδηγούς με διαφορετικές στρατηγικές και ρυθμίσεις.

Μετά τον αγώνα, υπάρχει το post-event πρόγραμμα [8], όπου η ομάδα συγκρίνει το εικονικό με το πραγματικό αποτέλεσμα και εξετάζει πώς να βελτιωθεί για τα επόμενα Grand Prix.

Συνεπώς, πριν ξεκινήσει η πρώτη ελεύθερη δοκιμή κάποιου αγώνα, οι ομάδες έχουν ήδη τρέξει εκατοντάδες χιλιάδες σενάρια αγώνα και κατατακτηρίων που λαμβάνουν υπόψη πιθανά Pit Stop και απώλειες χρόνου, φθορά ελαστικών, καιρικές μεταβλητές και σενάρια αλλαγής κατάστασης πίστας. Αυτές οι προσομοιώσεις καθορίζουν όχι μόνο τη στρατηγική του αγώνα αλλά και το τι δεδομένα πρέπει να συλλεχθούν την Παρασκευή και το Σάββατο στα FP1, FP2 και FP3.

1.2.5 Η αξία των δεδομένων

Σε ένα περιβάλλον όπου ο χρόνος στην πίστα είναι περιορισμένος και οι κανονισμοί περιορίζουν τις δοκιμές, τα δεδομένα είναι ο πιο κρίσιμος πόρος. Δεν υπάρχει η δυνατότητα να επαναληφθεί ένας γύρος, κάθε ευκαιρία συλλογής δεδομένων είναι μοναδική. Η σωστή ανάλυση και αξιοποίηση των δεδομένων από την πίστα και τους προσομοιωτές μπορεί να καθορίσει την έκβαση ενός πρωταθλήματος [5].

1.2.6 Πηγή Δεδομένων της Εργασίας

Η πρόσβαση στα δεδομένα των αγώνων της Formula 1 για την παρούσα διπλωματική πραγματοποιείται μέσω της βιβλιοθήκης FastF1 της Python. Η συγκεκριμένη βιβλιοθήκη παρέχει ένα πλήρες API σε python για την άντληση δεδομένων από αγώνες F1, περιλαμβάνοντας δεδομένα τηλεμετρίας, χρόνους γύρων, χρήση ελαστικών κ.α..

Η αναλυτική παρουσίαση της βιβλιοθήκης και του τρόπου ενσωμάτωσής της στην εφαρμογή γίνεται στο Κεφάλαιο 2, όπου παρουσιάζονται οι βασικές λειτουργίες της.

1.3 Κίνητρο

Στα προηγούμενα υποκεφάλαια παρουσιάστηκαν οι βασικές έννοιες που αφορούν τους αγώνες της Formula 1 και τη σημασία των δεδομένων. Είναι σαφές ότι η F1 αποτελεί ένα περιβάλλον όπου τα δεδομένα και η ανάλυση τους παίζουν καθοριστικό ρόλο στην απόδοση των ομάδων και των οδηγών.

Ο τεράστιος όγκος δεδομένων που παράγεται σε κάθε αγωνιστικό τριήμερο έχουν δημιουργήσει την ανάγκη για εύχρηστα εργαλεία ανάλυσης και οπτικοποίησης. Ενώ υπάρχουν βιβλιοθήκες και APIs που παρέχουν πρόσβαση σε δεδομένα για την F1, τα περισσότερα από αυτά απαιτούν γνώσεις προγραμματισμού και εμπειρία σε ανάλυση δεδομένων. Περιορίζοντας έτσι την αξιοποίησή τους από το ευρύ κοινό.

Η βιβλιοθήκη FastF1 της Python αποτελεί ένα πλήρες εργαλείο πρόσβασης σε τηλεμετρία και σε δεδομένα αγώνων. Ωστόσο, η χρήση της γίνεται αποκλειστικά μέσω κώδικα, γεγονός που καθιστά δύσκολη την αξιοποίησή της από άτομα χωρίς εμπειρία στον προγραμματισμό. Στην πράξη, αυτό σημαίνει ότι οι φίλοι του αθλήματος, οι δημοσιογράφοι ή ακόμα και οι φοιτητές που θέλουν να μελετήσουν τα δεδομένα της Formula 1 βρίσκονται αντιμέτωποι με ένα δύσκολο εμπόδιο.

Παράλληλα, οι υπάρχουσες λύσεις για την οπτικοποίηση δεδομένων F1 είναι περιορισμένες, είτε πρόκειται για εξειδικευμένα εργαλεία με συνδρομή, είτε για custom scripts που δεν απευθύνονται στο ευρύ κοινό. Δεν υπάρχει επαρκής αριθμός εφαρμογών που να προσφέρουν διαδραστική διεπαφή, εύκολη πρόσβαση σε δεδομένα και δυνατότητα ανάλυσης χωρίς τεχνικές γνώσεις.

Έτσι τα κενά στην απεικόνιση και την διαθεσιμότητά των δεδομένων αποτέλεσαν το βασικό κίνητρο για την ανάπτυξη της παρούσας διπλωματικής. Η ανάγκη για μια εύχρηστη πλατφόρμα που κάνει την οπτικοποίηση δεδομένων Formula 1 και την κατανόησή τους από μη ειδικούς χρήστες εύκολη.

1.4 Συνεισφορά

Η παρούσα διπλωματική εργασία στοχεύει να καλύψει όλα τα κενά που αναφέρθηκαν παραπάνω, μέσω της ανάπτυξης μιας διαδικτυακής εφαρμογής που αξιοποιεί τη βιβλιοθήκη FastF1 της Python. Η εφαρμογή σχεδιάστηκε με σκοπό να παρέχει ένα διαδραστικό και εύχρηστο περιβάλλον για την ανάλυση δεδομένων αγώνων, απευθυνόμενη τόσο σε χρήστες με εμπειρία ανάλυσης όσο και σε χρήστες που δεν διαθέτουν γνώσεις προγραμματισμού.

Η εφαρμογή ενσωματώνει δεδομένα από όλες τις διαδικασίες κατά την διάρκεια των αγωνιστικών τριημέρων, παρέχοντας οπτικοποιήσεις τηλεμετρίας που περιλαμβάνουν παραμέτρους όπως η ταχύτητα, το ποσοστό χρήσης του γκαζιού, η λειτουργία του DRS, οι στροφές ανά λεπτό (RPM), ο χρόνος διαφοράς από κάποιο αντίπαλο (delta time) και το ποσοστό χρήσης του φρένου. Μέσα από διαδραστικά γραφήματα μπορούν να αναλυθούν οι χρόνοι των οδηγών, οι τελικές ταχύτητες που ανέπτυξαν, η χρήση των ελαστικών κ.α.. Παράλληλα, η φιλική προς τον χρήστη γραφική διεπαφή επιτρέπει την πλοήγηση στα δεδομένα χωρίς την ανάγκη γνώσης ή ανάπτυξης κώδικα, γεγονός που διευρύνει σημαντικά το κοινό που μπορεί να χρησιμοποιήσει το εργαλείο.

Επιπλέον, η εφαρμογή διαθέτει ένα API που επιτρέπει την εξαγωγή των δεδομένων σε μορφή JSON, δίνοντας την δυνατότητα για περαιτέρω ανάλυση ή ενσωμάτωση σε άλλες πλατφόρμες.

Με αυτή την προσέγγιση, η εργασία στοχεύει να μειώσει το κενό ανάμεσα στα εργαλεία που απαιτούν προγραμματιστικές γνώσεις και στην ανάγκη για εύκολη πρόσβαση και οπτισκοποίηση στα δεδομένα της Formula 1. Η εφαρμογή μπορεί να αποτελέσει χρήσιμο εργαλείο για φοιτητές, δημοσιογράφους, ερευνητές, αναλυτές δεδομένων αλλά και για τους φίλους του αθλήματος που επιθυμούν να εξερευνήσουν λίγο περισσότερο τον κόσμο της F1 μέσα από δεδομένα και αναλύσεις.

1.5 Οργάνωση της Διπλωματικής

Η διπλωματική εργασία οργανώνεται σε επτά κεφάλαια, με στόχο να παρουσιαστεί με συστηματικό τρόπο τόσο το θεωρητικό υπόβαθρο όσο και η διαδικασία ανάπτυξης της εφαρμογής.

Στο Κεφάλαιο 1 είδαμε τις βασικές έννοιες που αποτελούν το πλαίσιο της εργασίας. Αρχικά, έγινε μια συνοπτική παρουσίαση της Formula 1 και της ιστορίας της, ενώ αναλύθηκαν οι διαδικασίες ενός τυπικού αγωνιστικού Σαββατοκύριακου. Στη συνέχεια, παρουσιάστηκε ο ρόλος και η σημασία των δεδομένων στους αγώνες F1. Ακολουθήσαν οι ενότητες που περιγράφουν το κίνητρο πίσω από την επιλογή του θέματος και τη συνεισφορά της παρούσας διπλωματικής.

Το Κεφάλαιο 2 αφορά τη βιβλιοθήκη FastF1, η οποία αποτελεί την κύρια πηγή δεδομένων για την εφαρμογή. Παρουσιάζονται οι βασικές λειτουργίες της, ο τρόπος άντλησης των δεδομένων καθώς, οι δυνατότητες που προσφέρει για ανάλυση και οπτικοποίηση και μερικά παραδείγματα χρήσης της.

Στο Κεφάλαιο 3 περιγράφονται οι γλώσσες και οι τεχνολογίες που χρησιμοποιήθηκαν για την ανάπτυξη της εφαρμογής. Για την υλοποίηση του Frontend και τον σχεδιασμό της διεπαφής χρήστη, χρησιμοποιήθηκαν οι τεχνολογίες React και TypeScript, οι οποίες συνέβαλλαν στη δημιουργία μιας δυναμικής, επεκτάσιμης και σύγχρονης web εφαρμογής. Η χρήση αυτών των τεχνολογιών επέτρεψε την ανάπτυξη επαναχρησιμοποιήσιμων components και την ενσωμάτωση διαδραστικών λειτουργιών που βελτιώνουν σημαντικά την εμπειρία του χρήστη. Από την άλλη, για την υλοποίηση του Backend, χρησιμοποιήθηκε η γλώσσα PHP, η οποία αλληλεπιδρά με τη βάση δεδομένων MySQL για την αποθήκευση, ανάκτηση και επεξεργασία των δεδομένων. Ενώ χρησιμοποιήθηκε και η γλώσσα Python για την άντληση και επεξεργασία των δεδομένων από την βιβλιοθήκη FastF1.

Στο Κεφάλαιο 4 γίνεται μια εκτενή αναφορά για τη σχεδίαση και την υλοποίηση της εφαρμογής. Παρουσιάζονται οι λειτουργικές απαιτήσεις, η αρχιτεκτονική της εφαρμογής με τη βοήθεια διαγραμμάτων ροής και ER καθώς και οι τεχνικές λεπτομέρειες υλοποίησης τόσο του backend όσο και του frontend.

Στο Κεφάλαιο 5 παρουσιάζεται αναλυτικά η εφαρμογή που αναπτύχθηκε. Μέσα από περιγραφές και οπτικά παραδείγματα, εξηγείται η λειτουργικότητα της πλατφόρμας, η διεπαφή χρήστη και οι βασικές δυνατότητες οπτικοποίησης δεδομένων Formula 1 που παρέχει.

Στο Κεφάλαιο 6 παρουσιάζεται η αξιολόγηση της εφαρμογής μέσω του System Usability Scale (SUS), όπου αναλύονται τα αποτελέσματα των απαντήσεων των χρηστών και εξάγονται χρήσιμα συμπεράσματα σχετικά με τη φιλικότητα και τη χρηστικότητα της εφαρμογής.

Τέλος, το Κεφάλαιο 7 περιλαμβάνει τα συμπεράσματα που προέκυψαν από την ανάπτυξη και χρήση της εφαρμογής, καθώς και προτάσεις για μελλοντικές επεκτάσεις.

Κεφάλαιο 2

Εισαγωγή στη βιβλιοθήκη FastF1

2.1 Εισαγωγή στη βιβλιοθήκη FastF1

Η βιβλιοθηκη FastF1 είναι μια ανοιχτού κώδικα βιβλιοθήκη της Python, σχεδιασμένη για την πρόσβαση, επεξεργασία και ανάλυση δεδομένων της Formula 1. Στόχος της είναι να παρέχει ανοιχτά δεδομένα της F1 σε προγραμματιστές, δημοσιογράφους και φίλους της Formula 1 [9].

Η βιβλιοθήκη προσφέρει πρόσβαση σε μια πληθώρα από δεδομένα για την F1 (τηλεμετρία, χρονομετρήσεις γύρων και αποτελέσματα) μέσω ειδικών API, ενώ τα δεδομένα που επιστρέφει είναι σε μορφή Pandas DataFrames, επιτρέποντας εύκολη ανάλυση και διαχείριση με τα εργαλεία της Python για διαχείριση δεδομένων. Παράλληλα, διαθέτει ενσωματωμένες λειτουργίες για γρήγορη παραγωγή γραφημάτων με τη χρήση του Matplotlib, κάτι που την καθιστά ιδιαίτερα χρήσιμη για την οπτικοποίηση δεδομένων.

Η βιβλιοθήκη έχει γίνει ιδιαίτερα δημοφιλής στην κοινότητα των data analysts και των φίλων της Formula 1, καθώς προσφέρει πρόσβαση σε λεπτομερή αγωνιστικά δεδομένα που μέχρι πρότινος ήταν δύσκολο να αποκτηθούν χωρίς εμπορικές πλατφόρμες.

2.2 Βασικές Δυνατότητες της βιβλιοθήκης FastF1

Η βιβλιοθήκη προσφέρει ένα ολοκληρωμένο σύνολο εργαλείων για την πρόσβαση και ανάλυση δεδομένων της Formula 1. Μέσα από τα ενσωματωμένα API που διαθέτει, επιτρέπει την ανάκτηση δεδομένων για κάθε διαδικασία, όπως χρόνους γύρων, επιδόσεις ανά τομέα, δεδομένα τηλεμετρίας, χρήση ελαστικών κ.α.. Η συμβολή του Ergast και του jolpica-f1 API δίνει στη βιβλιοθήκη τη δυνατότητα να αντλεί τόσο τρέχοντα όσο και ιστορικά δεδομένα και πληροφορίες για αγώνες, οδηγούς και ομάδες. Τα συγκεκριμένα APIs διαθέτουν ιστορικά δεδομένα και πληροφορίες για τους αγώνες, όπως τους οδηγούς που συμμετείχαν σε κάποιο αγώνα, ημερομηνίες διεξαγωγής των αγώνων κ.α. [9].

Εκτός από την τηλεμετρία γύρων, η βιβλιοθήκη παρέχει και πρόσβαση και σε άλλα δεδομένα, εξίσου σημαντικά για την ανάλυση της απόδοσης των ομάδων. Τα δεδομένα περιλαμβάνουν απο-

τελέσματα των διαδικασιών, πληροφορίες για τους γύρους που πραγματοποίησαν οι οδηγοί, της καιρικές συνθήκες, τη θέση των οδηγών στην σειρά εκκίνησης, τα ελαστικά που χρησιμοποιούνται καθώς και πληροφορίες για αγωνίστηκα συμβάντα, όπως κόκκινες σημαίες, αυτοκίνητο ασφαλείας και εικονικό αυτοκίνητο ασφαλείας. Με αυτή τη δυνατότητα παρέχεται μια πιο ολοκληρωμένη ανάλυση δεδομένων και χρήση παραμέτρων που επηρεάζουν την απόδοση στον αγώνα.

Ένα από τα βασικά χαρακτηριστικά της είναι η χρήση δομών Pandas DataFrame για την παρουσίαση των δεδομένων. Με αυτόν τον τρόπο, οι χρήστες μπορούν να εκμεταλλευτούν της δυνατότητες της βιβλιοθήκης Pandas, ενώ ταυτόχρονα έχουν πρόσβαση σε εξειδικευμένες συναρτήσεις και μεθόδους που έχουν σχεδιαστεί ειδικά για την επεξεργασία των συγκεκριμένων δεδομένων. Η δυνατότητα απευθείας ενσωμάτωσης με τη βιβλιοθήκη Matplotlib διευκολύνει σημαντικά τη δημιουργία γραφημάτων και την οπτικοποίησης των δεδομένων.

Ιδιαίτερη σημασία έχει και ο μηχανισμός caching που διαθέτει η FastF1, ο οποίος αποθηκεύει τοπικά τα δεδομένα που έχουν ανακτηθεί από τα API. Η λειτουργία αυτή μειώνει τον χρόνο εκτέλεσης των scripts και καθιστά τη βιβλιοθήκη αποδοτική ακόμη και σε περιπτώσεις ανάλυσης μεγάλου όγκου δεδομένων ή επαναλαμβανόμενων ερωτημάτων. Επιπλέον, επειδή είναι ανοιχτού κώδικα υπάρχει ενεργή κοινότητα χρηστών και προγραμματιστών που την ενημερώνει και την βελτιώνει συνεχώς.

2.3 Παραδείγματα Χρήσης της βιβλιοθήκης FastF1

Η χρήση της FastF1 ξεκινά με την εγκατάστασή της μέσω του διαχειριστή πακέτων pip:

```
pip install fastf1
```

Μετά την εγκατάσταση, ο χρήστης μπορεί να ανακτήσει δεδομένα από οποιοδήποτε διαδικασία, όπως έναν αγώνα ή τις κατατακτήριες.

2.3.1 Παράδειγμα 1

Στο παρακάτω παράδειγμα παρουσιάζεται πώς γίνεται η φόρτωση δεδομένων για έναν αγώνα και μια απλή οπτικοποίηση των χρόνων γύρων δύο οδηγών:

Κώδικας Python χρήσης της βιβλιοθήκης FastF1:

```
import fastf1
from fastf1 import plotting
import matplotlib.pyplot as plt

session = fastf1.get_session(2023, 'Monaco', 'R')
session.load()
```

```
# Επιλογή γύρων για δύο οδηγούς
laps_ver = session.laps.pick_driver('VER')
laps_lec = session.laps.pick_driver('LEC')

redbull_color = ("#4781D7")
ferrari_color = ("#ED1131")

# Δημιουργία γραφήματος με τους χρόνους γύρων
fig, ax = plt.subplots()
ax.plot(laps_ver['LapNumber'],
 ↪  laps_ver['LapTime'].dt.total_seconds(),
 ↪  color=redbull_color, label='VER')
ax.plot(laps_lec['LapNumber'],
 ↪  laps_lec['LapTime'].dt.total_seconds(),
 ↪  color=ferrari_color, label='LEC')

ax.set_xlabel("Αριθμός Γύρου")
ax.set_ylabel("Χρόνος Γύρου (δευτ.)")

ax.legend()

plt.title("Σύγκριση χρόνων γύρων – GP Μονακό 2023")
plt.show()
```

Σχήμα 2.1: Γράφημα σύγκριση χρόνων

Στο παράδειγμα αυτό, ανακτάμε τα δεδομένα από τον αγώνα στο Μονακό το 2023 και στη συνέχεια συγκρίνουμε τους χρόνους των γύρων για τους Verstappen (VER) και Leclerc (LEC). Αρχικά

γίνεται ανάκτηση όλων των γύρων για κάθε οδηγό και έπειτα δημιουργείται διάγραμμα γραμμής με το Matplotlib, όπου στον οριζόντιο άξονα εμφανίζεται ο αριθμός γύρου και στον κατακόρυφο ο χρόνος του γύρου σε δευτερόλεπτα. Με διαφορετικά χρώματα (μπλε για Red Bull, κόκκινο για Ferrari) παρουσιάζονται οι επιδόσεις των δύο οδηγών για καλύτερη η οπτική σύγκριση. Στο τέλος προστίθενται οι ετικέτες των αξόνων, το υπόμνημα και ο τίτλος για του γράφηματος.

2.3.2 Παράδειγμα 2 - Τηλεμετρία

Στο επόμενο παράδειγμα συγκρίνουμε τα δεδομένα ταχύτητας, ποσοστό χρήσης γκαζιού και πέδησης μεταξύ δύο οδηγών στον ταχύτερο γύρο τους κατά τη διάρκεια των κατατακτήριων. Η τηλεμετρία περιλαμβάνει πληροφορίες όπως ταχύτητα, χρήση γκαζιού και φρένων, χρήση DRS κ.α. επιτρέποντας πιο λεπτομερή ανάλυση σε συγκεκριμένους γύρους. Αυτό επιτρέπει στις ομάδες και τους οδηγούς να εντοπίσουν λάθη και διαφορετικές τακτικές στην προσέγγιση μιας στροφής, στο σημείο πέδησης κ.α..

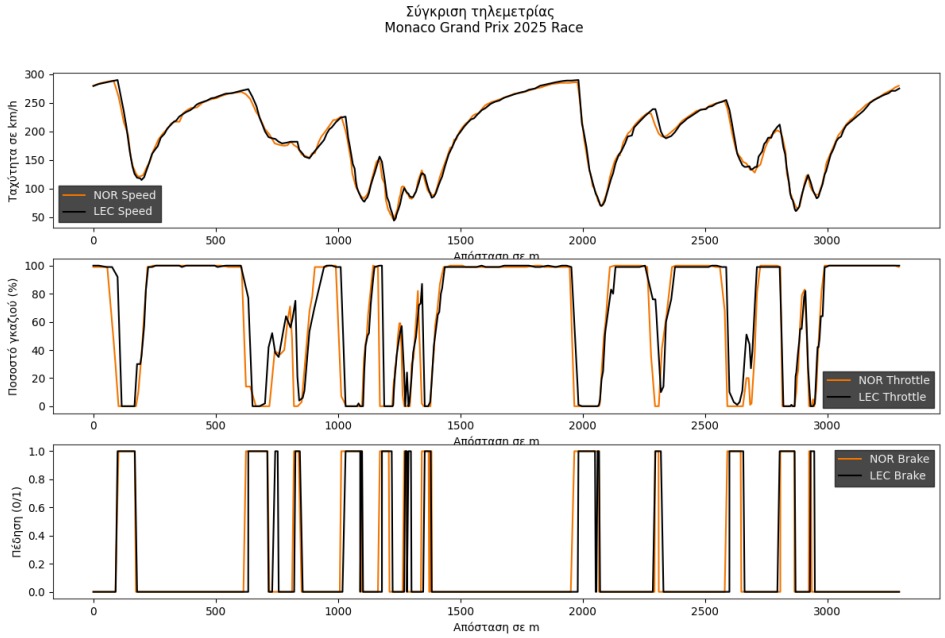
Κώδικας Python χρήσης της βιβλιοθήκης FastF1:

```
import fastf1
import matplotlib.pyplot as plt
import fastf1.plotting
import pandas as pd

fastf1.plotting.setup_mpl(mpl_timedelta_support=True,
 ↪  misc_mpl_mods=False,
                           color_scheme='fastf1')

# Φόρτωση δεδομένων για τον Αγώνα Μονακό 2023
session = fastf1.get_session(2025, 'Monaco', 'Q')
session.load()

# Επιλογή των ταχύτερων γύρων για δύο οδηγούς
nor_lap = session.laps.pick_drivers('NOR').pick_fastest()
lec_lap = session.laps.pick_drivers('LEC').pick_fastest()

# Ανάκτηση δεδομένων τηλεμετρίας για τους γύρους.
nor_tel = nor_lap.get_telemetry()
lec_tel = lec_lap.get_telemetry()

# Σχεδίαση γραφήματος ταχύτητας σε σχέση με την απόσταση
mclaren_color = ("#F47600")
ferrari_color = ("#000000")

fig, ax = plt.subplots(3, figsize=(10, 10),
 ↪  facecolor="white")
```

```
ax[0].plot(nor_tel['Distance'], nor_tel['Speed'],
 ↪  color=mclaren_color, label='NOR Speed')
ax[0].plot(lec_tel['Distance'], lec_tel['Speed'],
 ↪  color=ferrari_color, label='LEC Speed')

ax[1].plot(nor_tel['Distance'], nor_tel['Throttle'],
 ↪  color=mclaren_color, label='NOR Throttle')
ax[1].plot(lec_tel['Distance'], lec_tel['Throttle'],
 ↪  color=ferrari_color, label='LEC Throttle')


ax[2].plot(nor_tel['Distance'], nor_tel['Brake'],
 ↪  color=mclaren_color, label='NOR Brake')
ax[2].plot(lec_tel['Distance'], lec_tel['Brake'],
 ↪  color=ferrari_color, label='LEC Brake')

for a in ax:
    a.set_facecolor("white")
    a.tick_params(colors="black")
    a.xaxis.label.set_color("black")
    a.yaxis.label.set_color("black")
    a.title.set_color("black")


ax[0].set_xlabel('Απόσταση σε m')
ax[0].set_ylabel('Ταχύτητα σε km/h')

ax[1].set_xlabel('Απόσταση σε m')
ax[1].set_ylabel('Ποσοστό γκαζιού (%)')

ax[2].set_xlabel('Απόσταση σε m')
ax[2].set_ylabel('Πέδηση (0/1)')


ax[0].legend()
ax[1].legend()
ax[2].legend()

plt.suptitle(f"Σύγκριση τηλεμετρίας \n
 ↪  {session.event['EventName']} {session.event.year} Race",
 ↪  color="black")

plt.show()
```

Στο παραπάνω παράδειγμα παρουσιάζεται η τηλεμετρία για τις κατατακτήριες στον αγώνα του Μο-

νακό το 2025. Αρχικά γίνεται η φόρτωση των δεδομένων της συγκεκριμένης διαδικασίας και στη συνέχεια επιλέγεται ο ταχύτερος γύρος για κάθε οδηγό. Αφού βρούμε τον ταχύτερο γύρο για τους οδηγούς που μας ενδιαφέρει, Norris (NOR) και Leclerc (LEC), εξάγουμε την τηλεμετρία τους. Με το Matplotlib δημιουργούμε τρία διαγράμματα ως συνάρτηση της απόστασης στην πίστα: (1) Ταχύτητα (km/h), (2) Ποσοστό γκαζιού (%) , και (3) Πέδηση (0/1) (ως δυαδικό σήμα). Οι δύο οδηγοί έχουν διαφορετικά χρώματα (πορτοκαλί για McLaren, μαύρο για Ferrari), για καλύτερη σύγκριση. Στο τέλος προστίθενται ετικέτες αξόνων, υπόμνημα σε κάθε subplot και τίτλος στο γράφημα.



Σχήμα 2.2: Γράφημα τηλεμετρίας

2.4 Πλεονεκτήματα και Περιορισμοί της βιβλιοθήκης FastF1

Η βιβλιοθήκη FastF1 είναι ένα ισχυρό εργαλείο για την πρόσβαση και ανάλυση δεδομένων της Formula 1. Ένα από τα βασικά της πλεονεκτήματα είναι η πληθώρα των δεδομένων που παρέχει, καθώς αντλεί πληροφορίες απευθείας από επίσημες πηγές τηλεμετρίας και χρονομέτρησης. Η ενσωμάτωση με το Pandas επιτρέπει εύκολη και γρήγορη ανάλυση, ενώ ο συνδυασμός με το Matplotlib καθιστά την οπτικοποίηση των δεδομένων άμεση και προσαρμόσιμη στις ανάγκες κάθε ανάλυσης.

Παρά τα σημαντικά πλεονεκτήματά της, η βιβλιοθήκη παρουσιάζει και ορισμένους περιορισμούς. Ένας από τους κυριότερους είναι ότι απαιτεί τη χρήση κώδικα (σε Python), γεγονός που καθιστά τη βιβλιοθήκη ακατάλληλη σε χρήστες χωρίς εμπειρία στον προγραμματισμό. Επιπλέον, η πλήρης αξιοποίηση των δυνατοτήτων της προϋποθέτει εξοικείωση με βασικές έννοιες ανάλυσης δεδομένων, καθώς και με εργαλεία όπως το Pandas, κάτι που ενδέχεται να δυσκολέψει άτομα που δεν έχουν σχετική κατάρτιση.

Συνολικά, η FastF1 προσφέρει μια εξαιρετική βάση για την λήψη, επεξεργασία και οπτικοποίηση δεδομένων Formula 1, ωστόσο η χρήση της προϋποθέτει γνώσεις προγραμματισμού κάτι που την καθιστά άμεσα ακατάλληλη για το ευρύ κοινό.

Κεφάλαιο 3

Τεχνολογίες

3.1 Backend

3.1.1 PYTHON

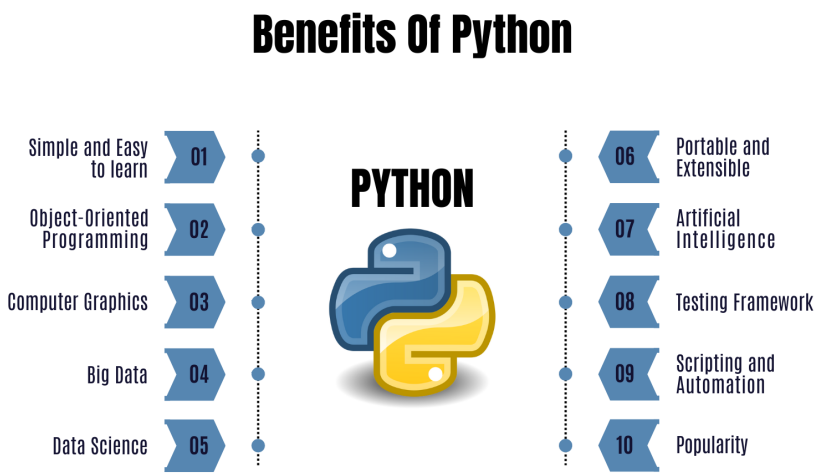
Η Python είναι μια υψηλού επιπέδου γλώσσα προγραμματισμού και ιδιαίτερα γνωστή για την απλότητα της. Δημιουργήθηκε από τον Guido van Rossum στα τέλη της δεκαετίας του 1980 και από τότε εξελίχθηκε σε μία από τις πιο δημοφιλείς γλώσσες παγκοσμίως, ειδικά στον χώρο της επιστήμης των δεδομένων, της μηχανικής μάθησης και της ανάπτυξης εφαρμογών. Η φιλοσοφία σχεδιασμού της Python δίνει έμφαση στην ευανάγνωστη και καθαρή σύνταξη, επιτρέποντας τη συγγραφή λογικού και καθαρού κώδικα τόσο για μικρά όσο και για μεγάλα έργα[10].

Η Python είναι ερμηνευόμενη γλώσσα, δηλαδή εκτελεί τον κώδικα γραμμή προς γραμμή, γεγονός που διευκολύνει στον εντοπισμό και τη διόρθωση λαθών. Η σύνταξή της είναι απλή και κοντά στην αγγλική γλώσσα, κάνοντάς την ιδανική για αρχάριους αλλά και για έμπειρους προγραμματιστές. Είναι ανοιχτού κώδικα και διανέμεται δωρεάν, επιτρέποντας τη χρήση της χωρίς περιορισμούς, ενώ η δια-λειτουργικότητας της εξασφαλίζει συμβατότητα με όλα τα βασικά λειτουργικά συστήματα.

Η Python, χάρη στην ευελιξία της, βρίσκει εφαρμογή σε ένα ευρύ φάσμα εργασιών, από την ανάπτυξη web εφαρμογών και τη δημιουργία αυτοματοποιημένων διαδικασιών, μέχρι την επεξεργασία και ανάλυση δεδομένων, την τεχνητή νοημοσύνη και γενικότερα την ανάπτυξη λογισμικού [11]. Διαθέτει χιλιάδες πρόσθετες βιβλιοθήκες που καλύπτουν σχεδόν κάθε ανάγκη που μπορεί να έχει ο προγραμματιστής, ενώ η ενεργή παγκόσμια κοινότητά της προσφέρει συνεχή υποστήριξη.

Η Python είναι επίσης εξαιρετική ως scripting γλώσσα για τη διασύνδεση διαφορετικών συστημάτων και υπηρεσιών. Χάρη στις βιβλιοθήκες της για διαχείριση αρχείων, δικτύων και APIs, μπορεί εύκολα να αντλεί δεδομένα από διάφορες πηγές (όπως web services, αρχεία καταγραφής ή απομακρυσμένους servers) να τα επεξεργάζεται και να τα μεταφέρει αυτόματα σε βάσεις δεδομένων για περαιτέρω ανάλυση ή χρήση [12].

Έτσι η Python, με την απλότητα και την προσαρμοστικότητά της, αποτελεί πολύ σημαντικό εργαλείο για κάθε προγραμματιστή, τόσο για την ανάπτυξη μικρών scripts όσο και για σύνθετες, μεγάλης κλίμακας εφαρμογές.



Σχήμα 3.1: Τα Πλεονεκτήματα της Python (Πηγή: Jaro Education)

3.1.2 MySQL

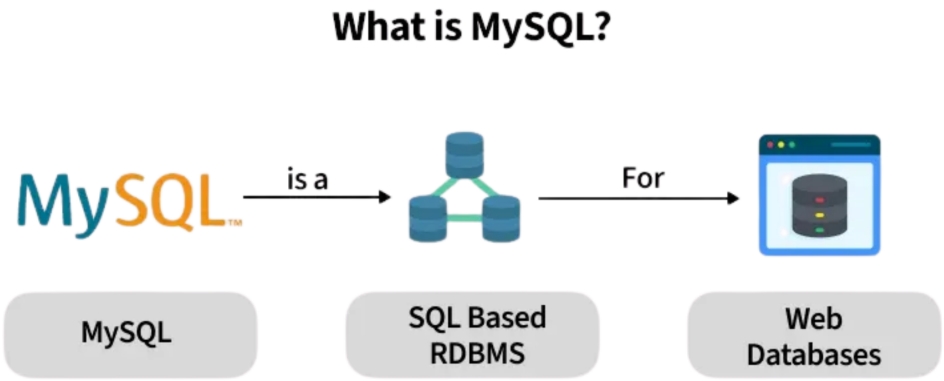
Η MySQL είναι ένα σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων (Relational Database Management System - RDBMS) ανοιχτού κώδικα, το οποίο χρησιμοποιεί τη γλώσσα SQL (Structured Query Language) για τη δημιουργία και διαχείριση βάσεων δεδομένων. Αναπτύχθηκε αρχικά από τη MySQL AB και σήμερα υποστηρίζεται και βελτιώνετε από την Oracle. Η MySQL αποτελεί μια από τις δημοφιλέστερες βάσεις δεδομένων παγκοσμίως, που χρησιμοποιείται τόσο σε μικρές εφαρμογές όσο και σε σύνθετα και μεγάλης κλίμακας πληροφοριακά συστήματα [13].

Η επιτυχία της MySQL αποδίδεται σε μεγάλο βαθμό στην ευκολία χρήσης, την αξιοπιστία, και την υψηλή απόδοση, χαρακτηριστικά που την καθιστούν ιδανική για απαιτητικές εφαρμογές όπως το Facebook, το Netflix και το Booking.com [14]. Ως σχεσιακή βάση δεδομένων, η MySQL αποθηκεύει τα δεδομένα σε πίνακες με γραμμές και στήλες, οργανωμένους σε σχήματα (schemas). Ένα σχήμα ορίζει πώς οργανώνονται και αποθηκεύονται τα δεδομένα και περιγράφει τις σχέσεις μεταξύ των διαφόρων πινάκων. Υποστηρίζει επίσης διάφορους τύπους δεδομένων, όπως κείμενο, αριθμούς, ημερομηνίες, ώρες, JSON και διανύσματα [15].

Ένα ακόμα βασικό πλεονέκτημα της MySQL είναι το γεγονός ότι είναι ελεύθερη στη χρήση, ενώ για εμπορική χρήση ή πρόσβαση σε επαγγελματικά χαρακτηριστικά, παρέχεται και ως εμπορική έκδοση μέσω της Oracle.

Επιπλέον, η MySQL έχει μια μεγάλη παγκόσμια κοινότητα που την περιβάλλει και διασφαλίζει τη συνεχή βελτίωση, γρήγορη επίλυση προβλημάτων και πληθώρα από παραδείγματα και βέλτιστες πρακτικές. Τέλος, η MySQL χαρακτηρίζεται επίσης από υψηλή απόδοση και αξιοπιστία, με δυνατότητα διαχείρισης μεγάλου όγκου δεδομένων και ταυτόχρονων συνδέσεων, παρέχοντας συνεχή λειτουργία ακόμη και σε απαιτητικές συνθήκες, χάρη στους μηχανισμούς για ελαχιστοποίηση του

κινδύνου απώλειας δεδομένων.



Σχήμα 3.2: Ορισμός της MySQL (Πηγή: Geeksforgeeks)

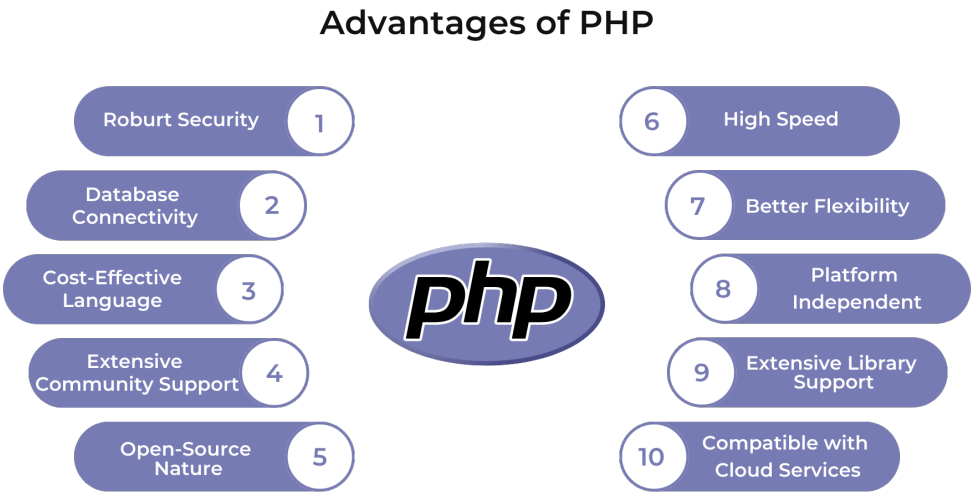
3.1.3 PHP

Η PHP (Hypertext Preprocessor) είναι μία ανοιχτού κώδικα γλώσσα προγραμματισμού για server-side scripting, που έχει σχεδιαστεί κυρίως για την ανάπτυξη διαδικτυακών εφαρμογών. Δημιουργήθηκε από τον Rasmus Lerdorf το 1994 και εξελίχθηκε σταδιακά σε ένα από τα πιο διαδεδομένα εργαλεία στον χώρο του web development. Με την πάροδο των χρόνων, η PHP έχει καθιερωθεί ως βασικό συστατικό πολλών γνωστών εφαρμογών, όπως το WordPress και το Joomla, ενώ χρησιμοποιείται ακόμη και από μεγάλες εταιρείες όπως το Facebook και τη Wikipedia. Στην αρχή, η συντομογραφία PHP σήμαινε Personal Homepage (Προσωπική Ιστοσελίδα), ενώ σήμερα, αποτελεί ακρωνύμιο του "Hypertext Preprocessor" (Προεπεξεργαστής Υπερκειμένου) [16].

Η επιτυχία της PHP οφείλεται σε συνδυασμό πολλών χαρακτηριστικών. Αρχικά, έχει το πλεονέκτημα της δια-λειτουργικότητας, δηλαδή μπορεί να τρέξει σε όλα τα μεγάλα λειτουργικά συστήματα (Windows, Linux, macOS) και να συνεργαστεί με τους δημοφιλέστερους web servers, όπως Apache και Nginx. Επίσης, υποστηρίζει ένα μεγάλο πλήθος βάσεων δεδομένων, τόσο σχεσιακών (MySQL, PostgreSQL, SQLite) [14] όσο και μη σχεσιακών (MongoDB).

Επιπρόσθετα, ένα σημαντικό χαρακτηριστικό της είναι η ευκολία εκμάθησης, γεγονός που την καθιστά προσβάσιμη σε νέους προγραμματιστές. Η σύνταξή της είναι απλή και κατανοητή, επιτρέποντας τη γρήγορη δημιουργία λειτουργικών σελίδων και εφαρμογών. Παράλληλα, η PHP είναι ανοιχτού κώδικα και διανέμεται δωρεάν, γεγονός που έχει ενισχύσει στην εξάπλωση της και έχει δημιουργήσει μια ενεργή παγκόσμια κοινότητα υποστήριξης [17].

Συνολικά, η PHP παραμένει μια αξιόπιστη και αποδοτική επιλογή για την ανάπτυξη web εφαρμογών κάθε κλίμακας, με δια-λειτουργικότητα, συμβατότητα με αρκετές βάσεις δεδομένων και την υποστήριξη μιας παγκόσμιας κοινότητας προγραμματιστών.



Σχήμα 3.3: Τα Πλεονεκτήματα της PHP

3.2 Frontend

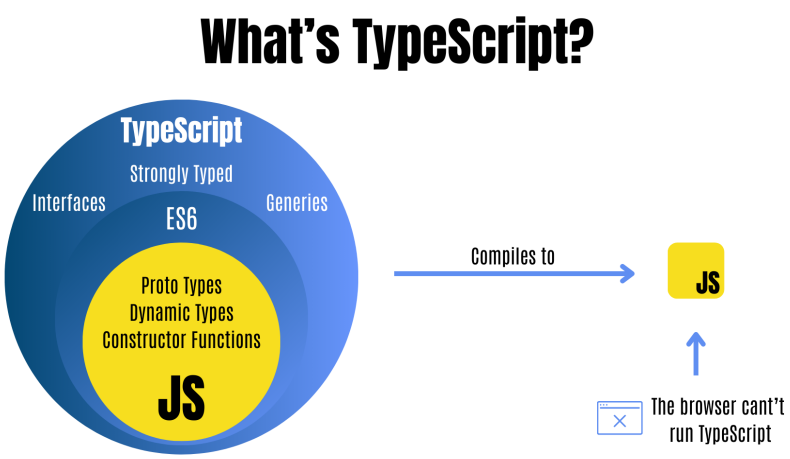
3.2.1 TypeScript

Η TypeScript είναι μια γλώσσα προγραμματισμού που δημιουργήθηκε από τη Microsoft και λειτουργεί ως υπερσύνολο της JavaScript. Αυτό σημαίνει ότι κάθε ένας κώδικας σε JavaScript είναι επίσης εκτελέσιμος και σε TypeScript. Όμως η TypeScript προσθέτει επιπλέον χαρακτηριστικά που κάνουν τον κώδικα πιο αξιόπιστο και εύκολα επεκτάσιμο σε μεγάλης κλίμακας εφαρμογές [18].

Η JavaScript αρχικά σχεδιάστηκε για να προσθέτει διαδραστικότητα σε ιστοσελίδες με περιορισμένη δυνατότητα ελέγχου τύπων. Παρόλα αυτά με την πάροδο του χρόνου εξελίχθηκε σημαντικά και παρέμεινε μια δυναμικά τυποποιημένη γλώσσα (δηλαδή οι μεταβλητές δεν χρειάζεται να δηλωθούν με συγκεκριμένο τύπο και μπορούν να αλλάζουν τύπο κατά την εκτέλεση), γεγονός που μπορεί να οδηγήσει εύκολα σε σφάλματα που εμφανίζονται μόνο κατά την εκτέλεση του κώδικα.

Παράδειγμα κώδικα JavaScript:

```
let example = "Hello, world!";
console.log(typeof example); // "string"
```



Σχήμα 3.4: Η TypeScript ως υπερσύνολο της JavaScript (Πηγή: Openxcell)

```
example = 42;
console.log(typeof example); // "number"
```

Η TypeScript έρχεται να καλύψει αυτό το κενό, προσφέροντας στατική τυποποίηση και μηχανισμό ελέγχου τύπων κατά τη μεταγλώττιση. Μέσα από αυτόν τον μηχανισμό, ο προγραμματιστής μπορεί να εντοπίσει σφάλματα ή λανθασμένες μεταβλητές πριν καν εκτελέσει τον κώδικα. Για παράδειγμα, αν κάποιος προσπαθήσει να χρησιμοποιήσει μια ιδιότητα ενός αντικειμένου με λανθασμένο όνομα (π.χ. obj.heigth αντί για obj.height), η JavaScript θα αγνοήσει το πρόβλημα και θα αποδώσει undefined, οδηγώντας πιθανόν σε περίεργα αποτελέσματα, ενώ από την άλλη η TypeScript θα επιστρέψει μήνυμα σφάλματος[19].

Μια άλλη σημαντική διαφορά είναι η δυνατότητα του ορισμού τύπων. Στην TypeScript, ο προγραμματιστής μπορεί να ορίσει εκ των προτέρων ότι μια μεταβλητή (είναι για παράδειγμα, αριθμός (number) και όχι συμβολοσειρά (string)) [20]. Αυτό περιορίζει την πιθανότητα λογικών σφαλμάτων. Παράλληλα, η TypeScript υποστηρίζει interfaces, enums, literal types, type unions και άλλες προηγμένες δομές που δεν υπάρχουν εξ ορισμού στην JavaScript, επιτρέποντας στον προγραμματιστή να ορίσει με μεγαλύτερη ακρίβεια τα δεδομένα.

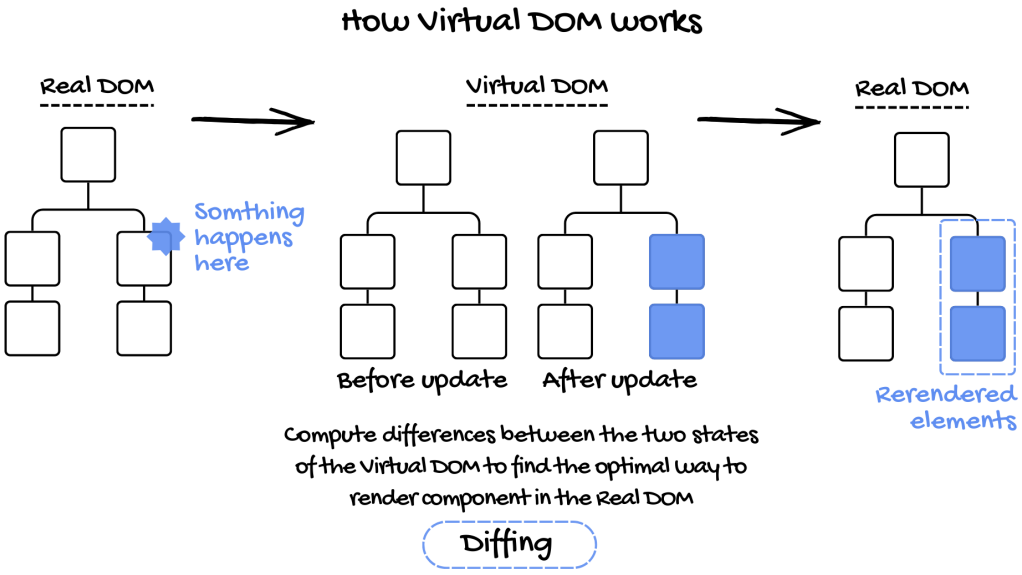
Είναι επίσης σημαντικό να σημειωθεί ότι η TypeScript δεν αλλάζει καθόλου τη συμπεριφορά εκτέλεσης του JavaScript κώδικα. Αντίθετα, μεταγλωττίζεται σε JavaScript, η οποία είναι συμβατή με όλους τους browsers, αλλά και με άλλα περιβάλλοντα, όπως το Node.js. Αυτό σημαίνει πως η TypeScript λειτουργεί αποκλειστικά στο στάδιο της ανάπτυξης της εφαρμογής [20].

Η γλώσσα έχει πλέον καθιερωθεί ως κύριο εργαλείο σε μεγάλα frontend frameworks και βιβλιοθήκες, όπως το React, το Angular, ενώ χρησιμοποιείται ευρέως και στο backend μέσω Node.js και άλλων frameworks.

3.2.2 React

Το React είναι μία βιβλιοθήκη JavaScript για την κατασκευή δυναμικών γραφικών διεπαφών χρήστη (UI), που αναπτύχθηκε από την ομάδα της Facebook το 2011 και δημοσιεύθηκε ως λογισμικό ανοιχτού κώδικα το 2013. Η ανάγκη για επαναχρησιμοποιήσιμα στοιχεία διεπαφής σε περιβάλλοντα με μεγάλη αλληλεπίδραση χρηστών, όπως το Facebook και το Instagram, αποτέλεσε τον βασικό λόγο ύπαρξής του [21].

Σε αντίθεση με την παραδοσιακή JavaScript, όπου η χειροκίνητη ενημέρωση του DOM είναι χρονοβόρα και συχνά προκαλεί χαμηλή απόδοση, το React εισήγαγε το εικονικό DOM (Virtual DOM) [22] , έναν ελαφρύ μηχανισμό που δημιουργεί μια εσωτερική αναπαράσταση της σελίδας. Όταν τα δεδομένα (states) μιας εφαρμογής αλλάζουν, το React συγκρίνει το νέο virtual DOM με το προηγούμενο και ενημερώνει μόνο τα απαραίτητα σημεία του πραγματικού DOM, βελτιστοποιώντας την απόδοση και μειώνοντας τις περιττές ανανεώσεις.



Σχήμα 3.5: Τρόπος λειτουργίας του Virtual DOM

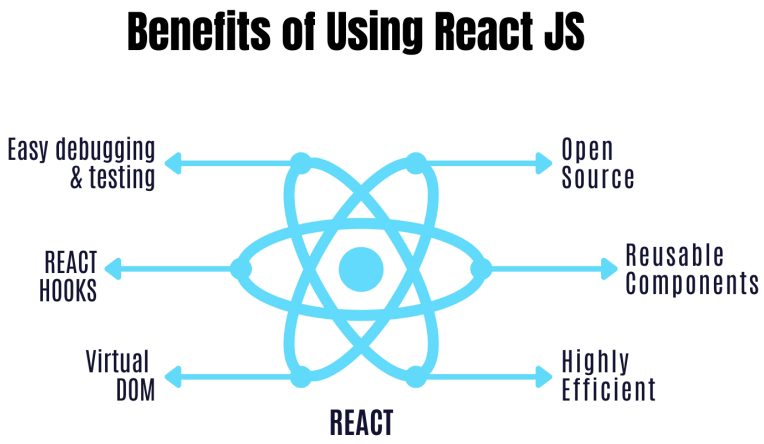
Το React βασίζεται σε δηλωτικό προγραμματισμό και τη χρήση του JSX, μιας σύνταξης που επιτρέπει στον προγραμματιστή να γράφει HTML κώδικα στο ίδιο αρχείο με τον κώδικα JavaScript. Με αυτόν τον τρόπο, η δομή και η λογική του UI συνδυάζονται σε μία μορφή, διευκολύνοντας την ανάπτυξη, την κατανόηση και τη συντήρηση του κώδικα.

Βασικό συστατικό του React είναι τα Components, δηλαδή επαναχρησιμοποιήσιμα κομμάτια UI με δική τους λογική, κατάσταση και εμφάνιση. Η κάθε εφαρμογή React αποτελείται από πλήθος τέτοιων components που συνεργάζονται, σχηματίζοντας μια αρθρωτή και επεκτάσιμη δομή. Αυτή η αρχιτεκτονική διευκολύνει την ανεξάρτητη επεξεργασία και επαναχρησιμοποίηση τμημάτων του UI.

Το React υποστηρίζει μοντέλο μονόδρομης ροής δεδομένων (unidirectional data flow). Τα δεδομένα περνούν από τα "parent" components προς τα "child", κάνοντας τη ροή προβλέψιμη και εύκολα εντοπίσιμη σε περίπτωση σφαλμάτων. Οι αλλαγές στο state ενός component προκαλούν αυτόματη επανασχεδίαση μόνο εκείνων των τμημάτων του UI που επηρεάζονται και όχι ολόκληρης της εφαρμογής [18].

Παρότι το React δεν είναι framework, καθώς επικεντρώνεται μόνο στο view layer της εφαρμογής, μπορεί να συνδυαστεί με άλλες βιβλιοθήκες ή εργαλεία για την υποστήριξη routing, global state management και API interaction, μετατρέποντας το σε πλήρες εργαλείο ανάπτυξης front-end εφαρμογών.

Η μεγάλη απήχηση του React οφείλεται επίσης στην ευκολία εκμάθησης, ιδίως για προγραμματιστές που ήδη γνωρίζουν JavaScript, στην ευελιξία και επεκτασιμότητά της, στην διαθεσιμότητα ενός μεγάλου πλήθους βιβλιοθηκών καθώς και στην υποστήριξη από μια ενεργή κοινότητα. Εξαιτίας αυτών των χαρακτηριστικών, χρησιμοποιείται σήμερα από πληθώρα επιχειρήσεων και οργανισμών σε παγκόσμιο επίπεδο, με παραδείγματα όπως η Netflix, το Airbnb και η Meta [19].



Σχήμα 3.6: Τα πλεονεκτήματα του React

3.2.3 JSX (JavaScript XML)

Η JSX (JavaScript XML) είναι μια επέκταση της σύνταξης της JavaScript, η οποία δημιουργήθηκε από τη Meta (Facebook) για χρήση με το React. Ουσιαστικά, πρόκειται για έναν πιο κατανοητό και ευανάγνωστο τρόπο γραφής των στοιχείων του React, που διαφορετικά θα έπρεπε να δημιουργούνται μέσω της συνάρτησης React.createElement(). [23]

Ένα σημαντικό χαρακτηριστικό της JSX είναι ότι τα στοιχεία της δεν είναι εντολές (statements) αλλά εκφράσεις (expressions). Αυτό σημαίνει ότι μπορούμε να τα αποθηκεύσουμε σε μεταβλητές,

να τα περάσουμε ως ορίσματα σε συναρτήσεις ή ακόμα και να τα χρησιμοποιήσουμε μέσα σε άλλα JSX στοιχεία. Η σύνταξή της θυμίζει αρκετά HTML, κάτι που κάνει τη συγγραφή και ανάγνωση του κώδικα πολύ πιο ευανάγνωστη, ειδικά όταν πρόκειται για πολύπλοκα UI.

Για παράδειγμα, ο κώδικας χωρίς JSX θα ήταν:

```
const element = React.createElement("h1", null, "Γειά σου
↪  JSX");
```

Ενώ με JSX, ο ίδιος κώδικας γίνεται πολύ πιο απλός και κατανοητός:

```
const element = <h1>Γειά σου JSX</h1>;
```

Επιπλέον, μπορούμε να χρησιμοποιήσουμε JSX για να δημιουργήσουμε και React components. Για παράδειγμα:

```
function Greeting() {
  return <h1>Καλώς ήρθες στη JSX!</h1>;
}
const element = <Greeting />;
```

Με αυτόν τον τρόπο, η JSX δεν περιορίζεται μόνο στη δημιουργία στοιχείων που μοιάζουν με HTML, αλλά προσφέρει και έναν εύχρηστο μηχανισμό για την αξιοποίηση επαναχρησιμοποιήσιμων components, τα οποία αποτελούν θεμελιώδες χαρακτηριστικό του React.

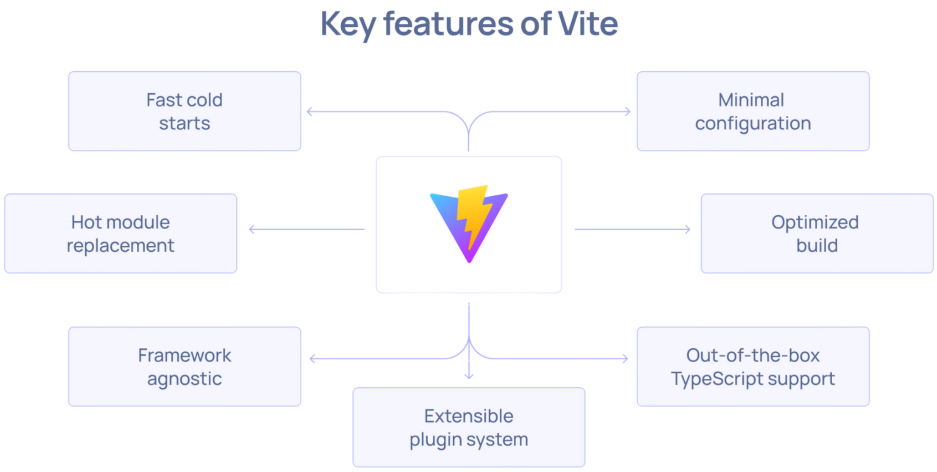
3.2.4 Vite

Το Vite είναι ένα σύγχρονο εργαλείο ανάπτυξης (build tool) για εφαρμογές ιστού, σχεδιασμένο να απλοποιήσει και να επιταχύνει την ανάπτυξη μιας εφαρμογής. Δημιουργήθηκε από τον Evan You, δημιουργό του Vue.js, με σκοπό να λύσει τα προβλήματα των υπάρχων εργαλείων, όπως το αργό φόρτωμα του development server και τις μεγάλες καθυστερήσεις στις αλλαγές κώδικα [24].

Ένα εργαλείο ανάπτυξης (build tool) είναι απαραίτητο για να διαχειρίζεται και να μετατρέπει τον κώδικα σε κατάλληλη μορφή για τον browser. Αυτό περιλαμβάνει διάφορες εργασίες όπως η μεταγλώττιση της JavaScript ή TypeScript σε κώδικα που υποστηρίζουν όλοι οι browsers, η συμπίεση των αρχείων για ταχύτερη φόρτωση κ.α.. Χωρίς ένα αποδοτικό build tool, η ανάπτυξη μιας εφαρμογής ιστού θα ήταν πιο αργή και πολύπλοκη, ιδιαίτερα σε μεγάλες εφαρμογές.

Μερικές από της σημαντικότερες δυνατότητες του Vite είναι η ταχύτατη εκκίνηση του server ακόμη και σε μεγάλες εφαρμογές, χάρη στη χρήση native ES modules που επιτρέπουν την άμεση εξυπηρέτηση των αρχείων, χωρίς αρχικό bundling της εφαρμογής. Επίσης, η λειτουργία Hot Module Replacement (HMR) προσφέρει την άμεση εμφάνιση των αλλαγών που γίνονται στον κώδικα κατευθείαν στον browser, χωρίς ανανέωση της σελίδας και με διατήρηση της τρέχουσας κατάστασης της εφαρμογής. Τέλος, κατά την διαδικασία παραγωγής του τελικού build, το Vite συνδυάζει την ταχύτητα του esbuild με την ευελιξία του Rollup, δημιουργώντας βελτιστοποιημένα αρχεία εξασφαλίζοντας υψηλές επιδόσεις στην τελική εφαρμογή [25].

Συνοψίζοντας, το Vite είναι ένα απαραίτητο εργαλείο για την ανάπτυξη και το χτίσιμο εφαρμογών, προσφέροντας έναν αποδοτικό τρόπο εργασίας που μειώνει την πολυπλοκότητά και αυξάνει την παραγωγικότητα του προγραμματιστή.



Σχήμα 3.7: Τα χαρακτηριστικά του Vite (Πηγή: Hygraph)

Κεφάλαιο 4

Σχεδίαση και Υλοποίηση του F1DataPlots

4.1 Λειτουργικές Απαιτήσεις

Οι λειτουργικές απαιτήσεις περιγράφουν λεπτομερώς τις δυνατότητες της εφαρμογής και τον τρόπο που αυτές ανταποκρίνονται στις ανάγκες των χρηστών. Για την εφαρμογή «F1DataPlots», οι απαιτήσεις περιλαμβάνουν:

Σελίδες Πληροφόρησης Εφαρμογής: Η εφαρμογή θα περιλαμβάνει σελίδες πληροφόρησης, όπως το «Home» και το «API Documentation». Η αρχική σελίδα (Home) θα περιέχει βασικές πληροφορίες για την εφαρμογή, ενώ η σελίδα «API Documentation» θα περιέχει πληροφορίες για τους προγραμματιστές, ώστε να κατανοούν τον τρόπο με τον οποίο μπορούν να αξιοποιήσουν το API της εφαρμογής.

Επιλογή Αγώνων που Ενδιαφέρουν τον Χρήστη: Οι χρήστες θα έχουν τη δυνατότητα να επιλέξουν τους αγώνες που τους ενδιαφέρουν για ανάλυση. Με την επιλογή ενός αγώνα, θα εμφανίζονται οι διαθέσιμες διαδικασίες (ελεύθερες δοκιμές, κατατακτήριες, αγώνας), από τις οποίες ο χρήστης θα μπορεί να επιλέξει εκείνη που τον ενδιαφέρει. Στη συνέχεια, θα προβάλλονται τα αντίστοιχα γραφήματα με τα διαθέσιμα δεδομένα της διαδικασίας.

Δημιουργία Διαδραστικών Γραφημάτων: Η εφαρμογή θα έχει τη δυνατότητα για δημιουργία διαδραστικών γραφημάτων για τα διαθέσιμα δεδομένα. Στην περίπτωση του Race Analysis, ο χρήστης θα μπορεί να επιλέξει την διαδικασία και το γράφημα που τον ενδιαφέρει, ενώ στο Telemetry Analysis θα μπορεί να επιλέξει έτος, αγώνα, οδηγό και γύρους για σύγκριση. Τα γραφήματα θα υποστηρίζουν δυναμική αλληλεπίδραση, όπως zoom in-out και εμφάνιση τιμών με hover.

Παραμετροποίηση των Γραφημάτων: Κάθε γράφημα θα μπορεί να παραμετροποιηθεί σύμφωνα με τις ανάγκες του χρήστη. Ο χρήστης θα έχει την δυνατότητα να αλλάξει τις παραμέτρους εμφάνισης, να φιλτράρει τα δεδομένα ή να προσαρμόσει την κλίμακα, έτσι ώστε να μπορεί εστιάσει στις πληροφορίες που τον ενδιαφέρουν περισσότερο.

Αποθήκευση των Δεδομένων των Γραφημάτων: Η εφαρμογή θα παρέχει την δυνατότητα αποθήκευσης των δεδομένων που απεικονίζονται σε κάθε γράφημα. Οι χρήστες θα μπορούν να κατεβάζουν τα δεδομένα σε μορφή αρχείων csv, ώστε να τα μπορούν να τα αξιοποιήσουν και εκτός της

εφαρμογής.

Αποθήκευση της Εικόνας του Γραφήματος: Πέρα από τα δεδομένα, οι χρήστες θα μπορούν να αποθηκεύουν και την εικόνα των γραφημάτων που δημιουργούν. Με αυτόν τον τρόπο θα διευκολύνεται η ενσωμάτωση των γραφημάτων σε αναφορές, παρουσιάσεις ή άλλο υποστηρικτικό υλικό.

Συγκριτική Ανάλυση Γύρων (Lap Comparison): Στην ενότητα Telemetry Analysis, οι χρήστες θα έχουν τη δυνατότητα να συγκρίνουν γύρους διαφορετικών οδηγών ή γύρους του ίδιου οδηγού στον ίδιο αγώνα. Η εφαρμογή εμφανίζει διαδραστικά γραφήματα τηλεμετρίας (π.χ. ταχύτητα, πέδηση, γκάζι, αλλαγές ταχυτήτων κ.α.), τα οποία βοηθούν στην σύγκριση των γύρων μεταξύ οδηγών.

Ενημέρωση με Βασικές Πληροφορίες Αγώνων: Στην αρχική σελίδα του Dashboard, οι χρήστες θα μπορούν να ενημερώνονται για τον επερχόμενο αγώνα και θα μπορούν να δουν συνοπτικά τις βαθμολογίες οδηγών και ομάδων. Με τον τρόπο αυτόν παρέχεται γρήγορη και εύκολη πρόσβαση σε κρίσιμες πληροφορίες της σεζόν.

4.2 Αρχιτεκτονική της Εφαρμογής

Όπως προαναφέρθηκε, ο στόχος της εφαρμογής «F1DataPlots» είναι να παρέχει στους χρήστες τη δυνατότητα να οπτικοποιούν και να αναλύουν δεδομένα από αγώνες Formula 1, με εύκολο και διαδραστικό τρόπο. Αρχικά, οι χρήστες, μπαίνοντας στην αρχική σελίδα, έχουν τη δυνατότητα να μεταβούν είτε στο Dashboard είτε στην τεκμηρίωση του API (API Documentation). Το Dashboard λειτουργεί ως μια κεντρική σελίδα διαχείρισης και πληροφόρησης, προσφέροντας συνοπτικά στοιχεία για το ερχόμενο Grand Prix, αλλά και τις βαθμολογίες οδηγών και ομάδων. Από εκεί, ο χρήστης μπορεί να επιλέξει ανάμεσα σε δύο βασικές λειτουργίες ανάλυσης: Race Analysis και Telemetry Analysis.

Στην ενότητα Race Analysis, οι χρήστες επιλέγουν τον αγώνα που τους ενδιαφέρει από μία λίστα διαθέσιμων αγώνων. Στη συνέχεια εμφανίζεται ένα περιβάλλον ανάλυσης, στο οποίο προβάλλονται οι διαθέσιμες διαδικασίες (ελεύθερες δοκιμές, κατατακτήριες, αγώνας) και τα αντίστοιχα διαθέσιμα γραφήματα. Ο χρήστης μπορεί να επιλέξει οποιοδήποτε γράφημα και να το προσαρμόσει σύμφωνα με τις ανάγκες του. Επιπλέον, παρέχεται η δυνατότητα αποθήκευσης της εικόνας του γραφήματος, καθώς και λήψης των δεδομένων του σε αρχείο csv.

Στην ενότητα Telemetry Analysis, οι χρήστες επιλέγουν έτος, αγώνα, διαδικασία και οδηγό. Στη συνέχεια μπορούν να επιλέξουν τον γύρο ή τους γύρους που θέλουν να συγκρίνουν. Η εφαρμογή εμφανίζει διαδραστικά γραφήματα τηλεμετρίας (π.χ. ταχύτητα, γκάζι, πέδηση, αλλαγές ταχυτήτων κ.α.), τα οποία βοηθούν στην κατανόηση της απόδοσης των οδηγών και των μονοθεσίων. Και σε αυτή την ενότητα, δίνεται η δυνατότητα παραμετροποίησης του γραφήματος και λήψης της τελικής εικόνας.

Όλες οι παραπάνω λειτουργίες είναι διαθέσιμες μέσω μιας φιλικής προς τον χρήστη γραφικής διεπαφής, η οποία έχει σχεδιαστεί με τρόπο ώστε να είναι εύκολη στην πλοήγηση και να υποστηρίζει

διαδραστική αλληλεπίδραση με τα δεδομένα. Παράλληλα, η εφαρμογή παρέχει και τεκμηρίωση του API, μέσω του οποίου οι προγραμματιστές μπορούν να δουν πώς να αλληλεπιδράσουν με τα δεδομένα της εφαρμογής. Τα αποτελέσματα του API εμφανίζονται σε μορφή JSON, ώστε να είναι εύκολα αξιοποιήσιμα και σε άλλα συστήματα ή αναλύσεις. Στο Σχήμα 4.1, παρουσιάζονται αναλυτικά, μέσα ενός διαγράμματος ροής, η σειρά των βημάτων και των λειτουργιών που ακολουθεί ο χρήστης στην εφαρμογή.

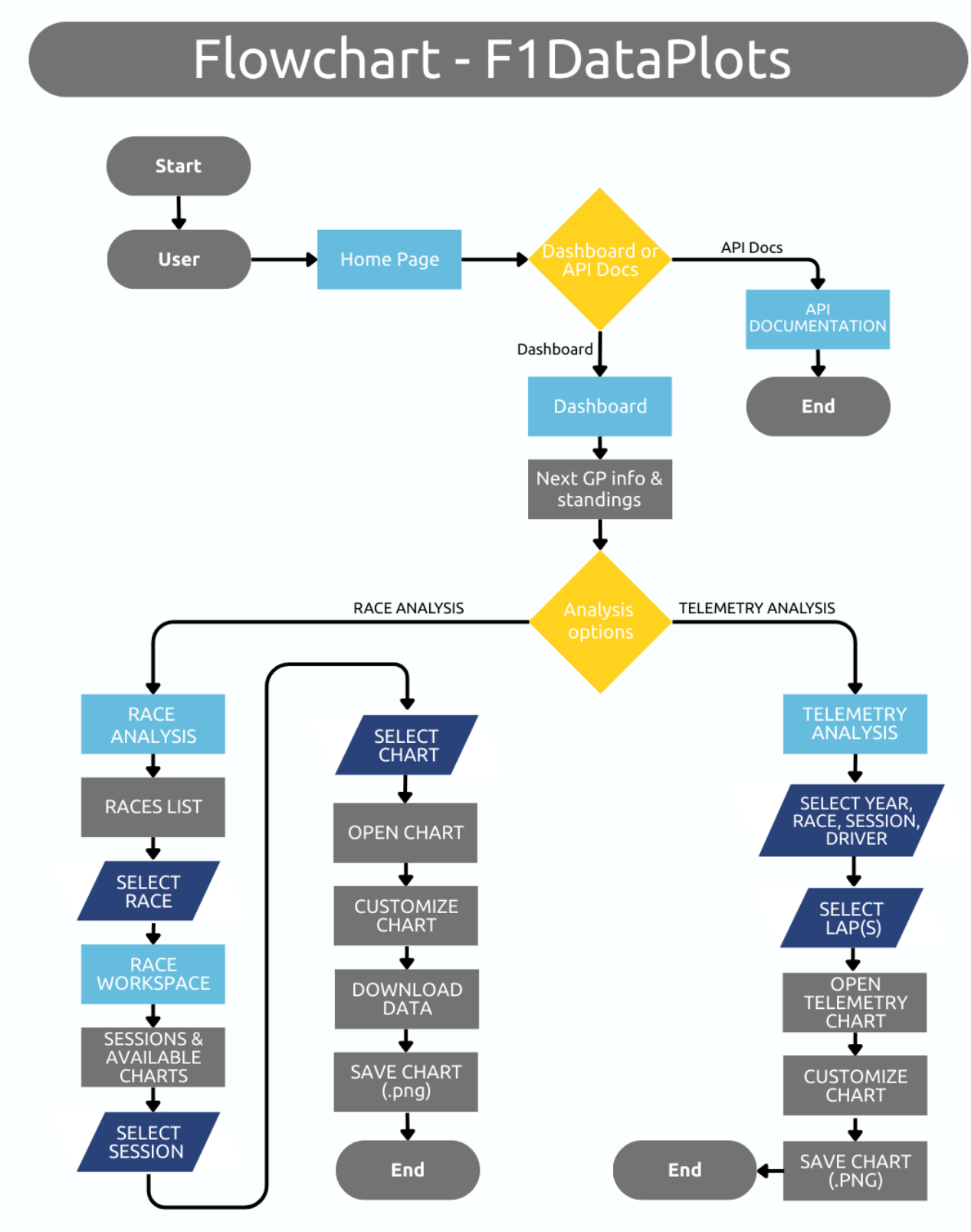
Η αρχιτεκτονική της εφαρμογής περιλαμβάνει τρία βασικά μέρη. Τον πυρήνα της αποτελεί το Web API, το οποίο λειτουργεί ως ενδιάμεσος κρίκος επικοινωνίας ανάμεσα στο Backend και στο Frontend και αναπτύχθηκε με την γλώσσα προγραμματισμού PHP. Το API διαχειρίζεται αιτήματα που σχετίζονται με τα δεδομένα και επιστρέφει τις κατάλληλες απαντήσεις στο Frontend.

Στο Backend χρησιμοποιείται μια σχεσιακή βάση δεδομένων για την αποθήκευση των πληροφοριών σχετικά με τους αγώνες, τους οδηγούς, τις διαδικασίες και τα δεδομένα για τα γραφήματα. Το API επικοινωνεί με τη βάση ώστε να αντλεί τα δεδομένα που απαιτούνται για τη δημιουργία των γραφημάτων.

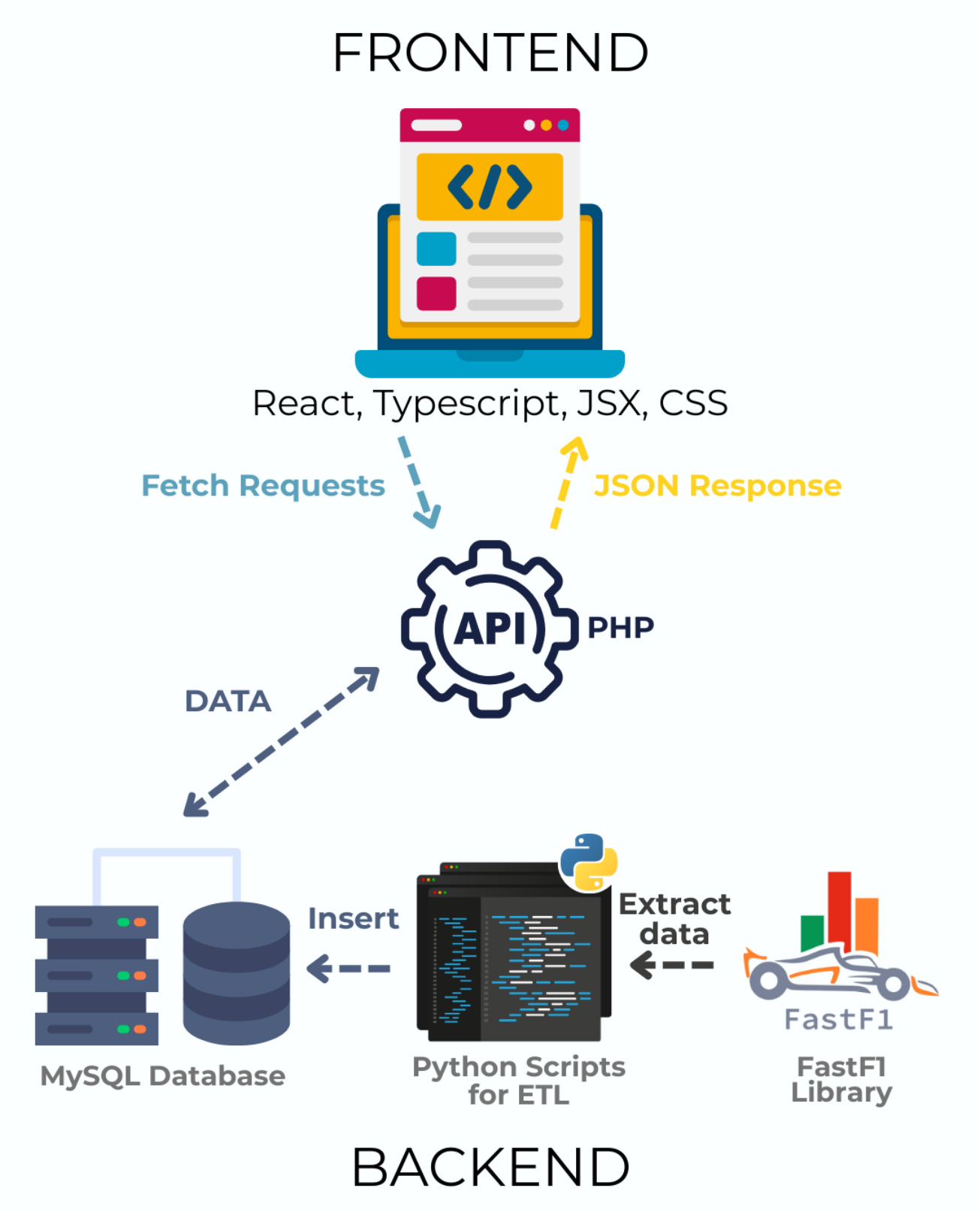
Παράλληλα με τα παραπάνω, υπάρχει και ένας ξεχωριστός μηχανισμός εισαγωγής δεδομένων που βασίζεται σε Python scripts. Τα scripts αυτά χρησιμοποιούν τη βιβλιοθήκη FastF1 ως πηγή δεδομένων και αναλαμβάνουν να αντλούν όλα τα απαραίτητα δεδομένα (όπως τηλεμετρία, χρόνοι γύρων κ.α.) να τα επεξεργάζονται και να τα καταχωρούν στη βάση δεδομένων. Η εκτέλεση των scripts γίνεται μέσω προγραμματισμένων εργασιών στον server ώστε η βάση να ενημερώνεται αυτόματα και σε τακτά χρονικά διαστήματα με τα πιο πρόσφατα δεδομένα.

Τέλος, στο Frontend, η γραφική διεπαφή και λειτουργικότητα έχουν υλοποιηθεί με React, Typescript, JSX και CSS. Το Frontend είναι υπεύθυνο για την εμφάνιση των γραφημάτων, την αλληλεπίδραση του χρήστη με τα δεδομένα, και την επικοινωνία με το Web API.

Η συνολική αρχιτεκτονική της εφαρμογής απεικονίζεται στο Σχήμα 4.2.



Σχήμα 4.1: Διάγραμμα Ροής του F1DataPlots



Σχήμα 4.2: Αρχιτεκτονική του F1DataPlots

4.3 Υλοποίηση του Backend

4.3.1 Βάση Δεδομένων

Όπως προαναφέρθηκε, η βάση δεδομένων της εφαρμογής είναι η MySQL και αποτελεί τον πύρινα τις εφαρμογής, καθώς εκεί αποθηκεύονται όλα τα δεδομένα που αντλούνται από την βιβλιοθήκη FastF1 και χρησιμοποιούνται αργότερα από τους χρήστες για την οπτικοποίησης τους. Έτσι, δημιουργήθηκε μία βάση στο server με όνομα 'f1dataplots_db' και στην συνέχεια με το λογισμικό HeidiSQL, έγινε η διαχείριση της.

Το HeidiSQL είναι ένα δωρεάν και ανοιχτού κώδικα λογισμικό διαχείρισης βάσεων δεδομένων. Έχει δημιουργηθεί για να προσφέρει μια εύκολη και φιλική εμπειρία χρήσης. Υποστηρίζει σύνδεση με διάφορες βάσεις δεδομένων, όπως MariaDB, MySQL, Microsoft SQL Server, PostgreSQL, SQLite, Interbase και Firebird. Μέσω της εφαρμογής, οι χρήστες έχουν τη δυνατότητα να επεξεργάζονται δεδομένα, να τροποποιούν τη δομή των πινάκων και να διαχειρίζονται διαφορετικά σχήματα βάσεων με ευκολία. Το HeidiSQL αναπτύχθηκε το 2002 από τον Ansgar Becker και παραμένει μέχρι και σήμερα ένα ιδιαίτερα διαδεδομένο εργαλείο [26].

Η βάση δεδομένων αποτελείται από 11 πίνακες, ορισμένοι από τους οποίους είναι αρκετά πολύπλοκοι λόγω του μεγάλου αριθμού μεταβλητών που περιλαμβάνουν. Συγκεκριμένα:

- Ο πίνακας `drivers` αποθηκεύει τα ονόματα και βασικά στοιχεία που αφορούν τους οδηγούς.
- Ο πίνακας `teams` περιέχει πληροφορίες για τις ομάδες.
- Ο πίνακας `circuits` καταγράφει λεπτομέρειες για κάθε πίστα που βρίσκεται στο πρόγραμμα της F1.
- Ο πίνακας `corners` περιλαμβάνει δεδομένα για τις στροφές κάθε πίστας.
- Ο πίνακας `races` περιέχει το πρόγραμμα των αγώνων και στοιχεία σχετικά με την τοποθεσία, την πίστα και το όνομα κάθε αγώνα.
- Ο πίνακας `sessions` καταγράφει τις διαδικασίες (π.χ. ελεύθερες δοκιμές, κατατακτήριες, αγώνας) κάθε αγώνα καθώς και την ώρα έναρξής τους.
- Ο πίνακας `laps` αποθηκεύει αναλυτικά δεδομένα για κάθε γύρο οδηγού, όπως συνολικό χρόνο, είδος ελαστικού, χρόνους στους επιμέρους τομείς της πίστας κ.α..
- Ο πίνακας `telemetry` περιλαμβάνει για κάθε γύρο μεγάλο όγκο δεδομένων τηλεμετρίας (π.χ. ταχύτητα, φρένα, γκάζι).
- Ο πίνακας `drivers_teams_season` συσχετίζει κάθε οδηγό με την ομάδα του για κάθε σεζόν.
- Ο πίνακας `results` καταγράφει τα αποτελέσματα όλων των διαδικασιών σε κάθε αγώνα.

Ο Πίνακας `drivers` (4.1) περιλαμβάνει επτά πεδία.

- Το πεδίο `driver_id` είναι το κύριο κλειδί του πίνακα και αποτελεί το μοναδικό αναγνωριστικό του κάθε οδηγού. Το πεδίο αυτό χρησιμοποιεί την εντολή `'auto_increment'`, που σημαίνει ότι το id δημιουργείται και εισάγεται με αυτόματο τρόπο στον πίνακα.

- Το πεδίο `driver_number`, αντιπροσωπεύει τον αριθμό που χρησιμοποιεί ο οδηγός στο μονοθέσιό του.
- Το πεδίο `first_name`, αντιπροσωπεύει το όνομα του οδηγού.
- Το πεδίο `last_name`, αντιπροσωπεύει το επώνυμο του οδηγού.
- Το πεδίο `'abbreviation'`, περιλαμβάνει την συντομογραφία με την οποία αναγνωρίζεται ο οδηγός σε πίνακες και γραφήματα
- Το πεδίο `'nationality'`, δηλώνει την εθνικότητα του οδηγού.
- Το πεδίο `'birthdate'`, αντιπροσωπεύει την ημερομηνία γέννησης του οδηγού.

Πίνακας 4.1: Πίνακας drivers της Βάσης Δεδομένων

Name	Datatype	Length/Set
driver_id	INT	11
driver_number	INT	50
first_name	VARCHAR	50
last_name	VARCHAR	50
abbreviation	VARCHAR	3
nationality	VARCHAR	50
birthdate	DATE	1

Ο Πίνακας `teams` teams (4.2) περιλαμβάνει τα παρακάτω τρια πεδία.

- Το πεδίο `'team_id'`, είναι το κύριο κλειδί του πίνακα και αποτελεί το μοναδικό αναγνωριστικό κάθε ομάδας. Το πεδίο αυτό χρησιμοποιεί την εντολή `'auto_increment'`, που σημαίνει ότι το id δημιουργείται και εισάγεται με αυτόματο τρόπο στον πίνακα.
- Το πεδίο `'name'`, περιέχει το όνομα της ομάδας.
- Το πεδίο `'color'`, αποθηκεύει τον κωδικό χρώματος που αντιπροσωπεύει την ομάδα (για χρήση σε γραφήματα ή οπτικοποιήσης).

Πίνακας 4.2: Πίνακας teams της Βάσης Δεδομένων

Name	Datatype	Length/Set
team_id	INT	-
name	VARCHAR	100
color	VARCHAR	7

Ο Πίνακας `circuits` (4.3) περιλαμβάνει πέντε πεδία.

- Το πεδίο `'circuit_id'` είναι το κύριο κλειδί του πίνακα και αποτελεί το μοναδικό αναγνωριστικό κάθε πίστας. Το πεδίο αυτό χρησιμοποιεί την εντολή `'auto_increment'`, που σημαίνει ότι το id δημιουργείται και εισάγεται με αυτόματο τρόπο στον πίνακα.
- Το πεδίο `'name'` περιέχει το όνομα της πίστας.
- Το πεδίο `'location'` αποθηκεύει την τοποθεσία (πόλη ή περιοχή) στην οποία βρίσκεται η πίστα.
- Το πεδίο `'country'` δηλώνει τη χώρα στην οποία βρίσκεται η πίστα.
- Το πεδίο `'length_km'` καταγράφει το συνολικό μήκος της πίστας σε χιλιόμετρα.

Πίνακας 4.3: Πίνακας circuits της Βάσης Δεδομένων

Name	Datatype	Length/Set
circuit_id	INT	11
name	VARCHAR	100
location	VARCHAR	100
country	VARCHAR	50
length_km	DECIMAL	5,5

Ο Πίνακας `corners` (4.4) περιλαμβάνει τα εξής οκτώ πεδία.

- Το πεδίο `'corner_id'` είναι το κύριο κλειδί του πίνακα και αποτελεί το μοναδικό αναγνωριστικό κάθε στροφής. Χρησιμοποιεί την εντολή `'auto_increment'`, ώστε η τιμή του να δημιουργείται και να εισάγεται αυτόματα.

- Το πεδίο `'circuit_id'` αποτελεί ξένο κλειδί που αναφέρεται στο πεδίο `'circuit_id'` του πίνακα `'circuits'`, προσδιορίζοντας σε ποια πίστα ανήκει η στροφή.
- Το πεδίο `'number'` περιέχει τον αριθμό της στροφής, όπως εμφανίζεται στην πραγματική πίστα.
- Το πεδίο `'letter'` χρησιμοποιείται για την αναπαράσταση περιπτώσεων όπου μια στροφή έχει υποδιαιρέσεις (π.χ. 1A, 1B).
- Το πεδίο `'x'` περιέχει την τιμή της συντεταγμένης στον άξονα X του σημείου της στροφής.
- Το πεδίο `'y'` περιέχει την τιμή της συντεταγμένης στον άξονα Y του σημείου της στροφής.
- Το πεδίο `'angle'` εκφράζει τη γωνία της στροφής σε μοίρες.
- Το πεδίο `'distance_m'` περιέχει την απόσταση (σε μέτρα) της στροφής από τη γραμμή εκκίνησης/τερματισμού της πίστας.

Πίνακας 4.4: Πίνακας corners της Βάσης Δεδομένων

Name	Datatype	Length/Set
corner_id	INT	-
circuit_id	INT	-
number	INT	-
letter	VARCHAR	2
x	FLOAT	-
y	FLOAT	-
angle	FLOAT	-
distance_m	FLOAT	-

Ο Πίνακας `races` (4.5) περιλαμβάνει τα εξής επτά πεδία.

- Το πεδίο `'race_id'`, είναι το κύριο κλειδί του πίνακα και αποτελεί το μοναδικό αναγνωριστικό κάθε αγώνα. Το πεδίο αυτό χρησιμοποιεί την εντολή `'auto_increment'`, που σημαίνει ότι το id δημιουργείται και εισάγεται με αυτόματο τρόπο στον πίνακα.
- Το πεδίο `'circuit_id'`, αποτελεί ξένο κλειδί του πεδίου `'circuit_id'` στον πίνακα `'circuit'` και δηλώνει την πίστα στην οποία διεξάγεται ο αγώνας.
- Το πεδίο `'name'`, περιέχει το όνομα του αγώνα.

- Το πεδίο `'round'`, αποθηκεύει τον αριθμό γύρου της σεζόν στον οποίο ανήκει ο αγώνας.
- Το πεδίο `'fullEventName'`, περιέχει την πλήρη ονομασία του GP.
- Το πεδίο `'date'`, περιέχει την ημερομηνία διεξαγωγής του αγώνα.
- Το πεδίο `'season_year'`, περιέχει το έτος της αγωνιστικής σεζόν.

Επιπλέον, στον πίνακα `races` έχουν οριστεί οι παρακάτω περιορισμοί μοναδικότητας για την αποφυγή διπλοεγγραφών:

- Το `unique_race_name_season`: εξασφαλίζει ότι το όνομα ενός αγώνα (`name`) δεν μπορεί να επαναληφθεί μέσα στην ίδια αγωνιστική σεζόν (`season_year`).
- Το `unique_race_round_season`: εξασφαλίζει ότι ο αριθμός γύρου (`round`) είναι μοναδικός για κάθε αγωνιστική σεζόν (`season_year`).

Με αυτόν τον τρόπο διασφαλίζεται η ακεραιότητα των δεδομένων, αποφεύγοντας την ύπαρξη αγώνων με το ίδιο όνομα ή τον ίδιο γύρο στην ίδια σεζόν.

Πίνακας 4.5: Πίνακας races της Βάσης Δεδομένων

Name	Datatype	Length/Set
race_id	INT	11
circuit_id	INT	11
name	VARCHAR	100
round	INT	11
fullEventName	VARCHAR	200
date	DATE	-
season_year	YEAR	-

Ο Πίνακας `sessions` (4.6) περιλαμβάνει τα εξής τέσσερα πεδία.

- Το πεδίο `'session_id'`, είναι το κύριο κλειδί του πίνακα και αποτελεί το μοναδικό αναγνωριστικό κάθε διαδικασίας. Χρησιμοποιεί την εντολή `'auto_increment'`, που σημαίνει ότι η τιμή του δημιουργείται και εισάγεται αυτόματα στον πίνακα.
- Το πεδίο `'race_id'`, αποτελεί ξένο κλειδί που αναφέρεται στο πεδίο `'race_id'` του πίνακα `'races'` και προσδιορίζει τον αγώνα στον οποίο ανήκει η διαδικασία.

- Το πεδίο `'session_type'`, περιέχει τον τύπο της συνεδρίας, όπως για παράδειγμα `Practice`, `Qualifying`, `Race` κ.α..
- Το πεδίο `'start_time'`, περιέχει την ημερομηνία και ώρα έναρξης της διαδικασίας.

Επιπλέον, στον πίνακα `sessions` έχει οριστεί ο περιορισμός `unique_session_type_per_race`, ο οποίος διασφαλίζει ότι για κάθε αγώνα (`race_id`) δεν μπορεί να υπάρχει περισσότερη από μία διαδικασία (`session_type`) με τον ίδιο τύπο.

Πίνακας 4.6: Πίνακας sessions της Βάσης Δεδομένων

Name	Datatype	Length/Set
session_id	INT	11
race_id	INT	11
session_type	ENUM	'FP1', 'FP2', 'FP3', 'Qualifying', 'Sprint Qualifying', 'Sprint' , 'Race'
start_time	DATETIME	-

Ο Πίνακας `laps` (4.7) περιλαμβάνει τα εξής τριάντα δυο πεδία.

- Το πεδίο `'lap_id'` είναι το κύριο κλειδί του πίνακα και αποτελεί το μοναδικό αναγνωριστικό κάθε γύρου. Χρησιμοποιεί την εντολή `'auto_increment'`, που σημαίνει ότι η τιμή του δημιουργείται και εισάγεται αυτόματα στον πίνακα.
- Το πεδίο `'session_id'` αποτελεί ξένο κλειδί που αναφέρεται στο πεδίο `'session_id'` του πίνακα `'sessions'` και δηλώνει τη διαδικασία στην οποία ανήκει ο γύρος.
- Το πεδίο `'driver_id'` αποτελεί ξένο κλειδί που αναφέρεται στο πεδίο `'driver_id'` του πίνακα `'drivers'` και προσδιορίζει τον οδηγό στον οποίο ανήκει ο γύρος.
- Το πεδίο `'team_id'` αποτελεί ξένο κλειδί που αναφέρεται στο πεδίο `'team_id'` του πίνακα `'teams'` και προσδιορίζει την ομάδα του οδηγού.
- Το πεδίο `'lap_time'` καταγράφει τον συνολικό χρόνο ολοκλήρωσης του γύρου (π.χ. 01:12.893). Για να διαπιστωθεί αν ο χρόνος γύρου έχει ακυρωθεί, γίνεται έλεγχος στο πεδίο `'deleted'`.
- Το πεδίο `'lap_number'` καταγράφει τον αριθμό του γύρου.
- Το πεδίο `'stint'` δηλώνει τον αριθμό του stint (διάστημα χρήσης ενός σετ ελαστικών).
- Το πεδίο `'pit_out_time'` περιέχει τον χρόνο της διαδικασίας κατά τον οποίο το μονοθέσιο εξήλθε από τα pits (π.χ. 00:57:17.421 δηλαδή 57 λεπτά από την έναρξη της διαδικασίας).
- Το πεδίο `'pit_in_time'` περιέχει τον χρόνο της διαδικασίας κατά τον οποίο το μονοθέσιο εισήλθε στα pits.

- Το πεδίο `'sector1_time'` καταγράφει τον χρόνο του πρώτου τομέα της πίστας σε δευτερόλεπτα.
- Το πεδίο `'sector2_time'` καταγράφει τον χρόνο του δεύτερου τομέα της πίστας σε δευτερόλεπτα.
- Το πεδίο `'sector3_time'` καταγράφει τον χρόνο του τρίτου τομέα της πίστας σε δευτερόλεπτα.
- Το πεδίο `'sector1_session_time'` καταγράφει τον χρόνο της διαδικασίας κατά τον οποίο ολοκληρώθηκε ο πρώτος τομέας.
- Το πεδίο `'sector2_session_time'` καταγράφει τον χρόνο της διαδικασίας κατά τον οποίο ολοκληρώθηκε ο δεύτερος τομέας.
- Το πεδίο `'sector3_session_time'` καταγράφει τον χρόνο της διαδικασίας κατά τον οποίο ολοκληρώθηκε ο τρίτος τομέας.
- Το πεδίο `'speed_i1'` καταγράφει την ταχύτητα στο speedtrap του πρώτου τομέα (km/h).
- Το πεδίο `'speed_i2'` καταγράφει την ταχύτητα στο speedtrap του δεύτερου τομέα (km/h).
- Το πεδίο `'speed_fl'` καταγράφει την ταχύτητα στη γραμμή τερματισμού (km/h).
- Το πεδίο `'speed_st'` καταγράφει την ταχύτητα στη μεγαλύτερη ευθεία της πίστας (km/h).
- Το πεδίο `'is_personal_best'` είναι μια σημαία που υποδεικνύει αν ο γύρος είναι ο επίσημα καλύτερος προσωπικός χρόνος του οδηγού. Εάν κάποιος γύρος είναι ταχύτερος αλλά έχει ακυρωθεί, το πεδίο αυτό παραμένει `false`.
- Το πεδίο `'compound'` περιέχει την ονομασία του τύπου ελαστικού που χρησιμοποιήθηκε (π.χ. SOFT, MEDIUM, HARD, INTERMEDIATE, WET).
- Το πεδίο `'tyre_life'` δηλώνει τον αριθμό γύρων που έχουν διανυθεί με το συγκεκριμένο σετ ελαστικών, συμπεριλαμβανομένων γύρων σε άλλες διαδικασίες αν πρόκειται για χρησιμοποιημένα ελαστικά.
- Το πεδίο `'fresh_tyre'` δηλώνει αν το σετ ελαστικών ήταν καινούργιο (δηλαδή `tyre_life = 0` στην αρχή του stint).
- Το πεδίο `'lap_start_time'` καταγράφει την χρονική στιγμή της διαδικασίας που ξεκίνησε ο γύρος.
- Το πεδίο `'lap_end_time'` καταγράφει την χρονική στιγμή της διαδικασίας που ολοκληρώθηκε ο γύρος.
- Το πεδίο `'timestamp'` καταγράφει την ακριβή χρονική στιγμή έναρξης του γύρου.
- Το πεδίο `'track_status'` περιέχει μία ακολουθία αριθμών που αντιπροσωπεύουν τις καταστάσεις της πίστας (π.χ. κίτρινη σημαία, κόκκινη σημαία) κατά τη διάρκεια του γύρου.
- Το πεδίο `'position'` καταγράφει τη θέση του οδηγού στο τέλος του γύρου (για FP1, FP2, FP3, sprint qualifying και qualifying μπορεί να είναι κενό).
- Το πεδίο `'deleted'` υποδεικνύει αν ο γύρος ακυρώθηκε από τους αγωνοδίκες (π.χ. για παραβίαση ορίων πίστας).
- Το πεδίο `'deleted_reason'` περιέχει τον λόγο ακύρωσης του γύρου.

- Το πεδίο `'fastf1_generated'` υποδεικνύει αν ο γύρος δημιουργήθηκε αυτόματα από το FastF1, συνήθως με περιορισμένες ή υπολογισμένες πληροφορίες.
- Το πεδίο `'is_accurate'` υποδεικνύει αν οι χρόνοι έναρξης και λήξης του γύρου έχουν συγχρονιστεί σωστά με τους υπόλοιπους γύρους.

Πίνακας 4.7: Πίνακας laps της Βάσης Δεδομένων

Name	Datatype	Length/Set
lap_id	INT	11
session_id	INT	11
driver_id	INT	11
team_id	INT	11
lap_time	TIME	-
lap_number	INT	11
stint	INT	11
pit_out_time	TIME	-
pit_in_time	TIME	-
sector1_time	TIME	-
sector2_time	TIME	-
sector3_time	TIME	-
sector1_session_time	TIME	-
sector2_session_time	TIME	-
sector3_session_time	TIME	-
speed_i1	FLOAT	-
speed_i2	FLOAT	-
speed_fl	FLOAT	-
speed_st	FLOAT	-
is_personal_best	BOOLEAN	-
compound	ENUM	'S','M','H','I','W'
tyre_life	INT	11
fresh_tyre	TINYINT	1
lap_start_time	TIME	-
lap_end_time	TIME	-
timestamp	DATETIME	-
track_status	VARCHAR	255
position	INT	11
deleted	TINYINT	1
deleted_reason	VARCHAR	255
fastf1_generated	TINYINT	1
is_accurate	TINYINT	1

Είναι αντιληπτό ότι ο παραπάνω πίνακας περιλαμβάνει μεγάλο αριθμό πεδίων, γεγονός που μπορεί να δυσκολέψει την κατανόηση και τη διαχείρισή του, ειδικά για κάποιον που βλέπει για πρώτη φορά τη βάση. Ωστόσο, η συγκέντρωση όλων αυτών των μεταβλητών σε έναν πίνακα εξασφαλίζει ότι κάθε εγγραφή (γύρος) περιέχει πλήρη και άμεσα διαθέσιμη πληροφορία χωρίς την ανάγκη πολλών ενώσεων (joins) κατά την ανάκτηση δεδομένων. Στην παρούσα εφαρμογή, όπου η ανάλυση των δεδομένων πραγματοποιείται αμέσως μετά τη λήξη των διαδικασιών, η προσέγγιση αυτή επιτρέπει την ταχύτερη και πιο αποδοτική επεξεργασία των δεδομένων.

Ο Πίνακας `telemetry` (4.8) περιλαμβάνει τα εξής δεκαπέντε πεδία.

- Το πεδίο `'telemetry_id'` είναι το κύριο κλειδί του πίνακα και αποτελεί το μοναδικό αναγνωριστικό κάθε εγγραφής τηλεμετρίας. Χρησιμοποιεί την εντολή `'auto_increment'`, που σημαίνει ότι η τιμή του δημιουργείται και εισάγεται αυτόματα στον πίνακα.
- Το πεδίο `'lap_id'` αποτελεί ξένο κλειδί που αναφέρεται στο πεδίο `'lap_id'` του πίνακα `'laps'` και συνδέει την εγγραφή τηλεμετρίας με τον αντίστοιχο γύρο.
- Το πεδίο `'timestamp'` περιέχει την ακριβή χρονική στιγμή (ημερομηνία και ώρα) καταγραφής του δείγματος τηλεμετρίας.
- Το πεδίο `'session_time'` περιέχει τον χρόνο της διαδικασίας κατά τον οποίο καταγράφηκε το δείγμα, μετρώντας από την έναρξη της.
- Το πεδίο `'time'` περιέχει τον σχετικό χρόνο γύρου στον οποίο αντιστοιχεί το δείγμα.
- Το πεδίο `'speed'` καταγράφει την ταχύτητα του μονοθεσίου σε km/h τη συγκεκριμένη χρονική στιγμή.
- Το πεδίο `'rpm'` καταγράφει τις στροφές ανά λεπτό (revolutions per minute) του κινητήρα.
- Το πεδίο `'n_gear'` αποθηκεύει την τρέχουσα σχέση στο κιβώτιο που είναι επιλεγμένη.
- Το πεδίο `'throttle'` αποτυπώνει το ποσοστό χρήσης του γκαζιού (σε ποσοστό %).
- Το πεδίο `'brake'` καταγράφει αν και πόσο εφαρμόζεται το φρένο (σε λογική τιμή 0 ή 1).
- Το πεδίο `'drs'` αποθηκεύει την κατάσταση του συστήματος DRS, δηλαδή αν είναι ενεργοποιημένο ή όχι.
- Το πεδίο `'distance'` περιέχει την απόσταση που έχει διανυθεί στον γύρο μέχρι το σημείο καταγραφής του δείγματος.
- Τα πεδία `'x'`, `'y'` και `'z'` αποθηκεύουν τις συντεταγμένες θέσης του μονοθεσίου στην πίστα, όπως καταγράφονται από το σύστημα τηλεμετρίας.

Πίνακας 4.8: Πίνακας τηλεμετρίας της Βάσης Δεδομένων

Name	Datatype	Length/Set
telemetry_id	INT	11
lap_id	INT	11
timestamp	DATETIME	3
session_time	TIME	3
time	TIME	3
speed	INT	11
rpm	INT	11
n_gear	INT	11
throttle	DECIMAL	5,2
brake	TINYINT	1
drs	TINYINT	1
distance	DOUBLE	-
x	DOUBLE	-
y	DOUBLE	-
z	DOUBLE	-

Ο Πίνακας `driver_team_season` (4.9) περιλαμβάνει τα εξής τέσσερα πεδία.

- Το πεδίο `'driver_team_id'` είναι το κύριο κλειδί του πίνακα και αποτελεί το μοναδικό αναγνωριστικό κάθε εγγραφής τηλεμετρίας. Χρησιμοποιεί την εντολή `'auto_increment'`, που σημαίνει ότι η τιμή του δημιουργείται και εισάγεται αυτόματα στον πίνακα.
- Το πεδίο `'driver_id'` αποτελεί ξένο κλειδί που αναφέρεται στο πεδίο `'driver_id'` του πίνακα `'drivers'`.
- Το πεδίο `'team_id'` αποτελεί ξένο κλειδί που αναφέρεται στο πεδίο `'team_id'` του πίνακα `'teams'`.
- Το πεδίο `'season_year'` περιέχει την χρονία συμμετοχής κάποιου οδηγού στην F1.

Πίνακας 4.9: Πίνακας συσχέτισης οδηγών και ομάδων για κάθε χρονιά της Βάσης Δεδομένων

Name	Datatype	Length/Set
driver_team_id	INT	11
driver_id	INT	11
team_id	INT	11
season_year	TIME	3

Ο Πίνακας `results` (4.10) περιλαμβάνει τα εξής τέσσερα πεδία.

- Το πεδίο `'result_id'` είναι το κύριο κλειδί του πίνακα και αποτελεί το μοναδικό αναγνωριστικό κάθε εγγραφής αποτελεσμάτων. Χρησιμοποιεί την εντολή `AUTO_INCREMENT`, που σημαίνει ότι η τιμή του δημιουργείται και εισάγεται αυτόματα στον πίνακα.
- Το πεδίο `'session_id'` αποτελεί ξένο κλειδί που αναφέρεται στο πεδίο `'session_id'` του πίνακα `'sessions'` και συνδέει το αποτέλεσμα με τη συγκεκριμένη διαδικασία.
- Το πεδίο `'driver_id'` αποτελεί ξένο κλειδί που αναφέρεται στο πεδίο `'driver_id'` του πίνακα `'drivers'` και προσδιορίζει τον οδηγό στον οποίο ανήκει το αποτέλεσμα.
- Το πεδίο `'team_id'` αποτελεί ξένο κλειδί που αναφέρεται στο πεδίο `'team_id'` του πίνακα `'teams'` και προσδιορίζει την ομάδα του οδηγού.
- Το πεδίο `'position'` περιέχει τη θέση του οδηγού κατά το τέλος της διαδικασίας, σύμφωνα με τα επίσημα αποτελέσματα.
- Το πεδίο `'classified_position'` περιέχει τη θέση κατάταξης του οδηγού, λαμβάνοντας υπόψη τυχόν ποινές ή άλλες τροποποιήσεις.
- Το πεδίο `'grid_position'` περιέχει τη θέση εκκίνησης του οδηγού στην εκκίνηση του αγώνα.
- Το πεδίο `'q1_time'` περιέχει τον χρόνο που σημείωσε ο οδηγός στο πρώτο μέρος των κατατακτηρίων δοκιμών (Q1).
- Το πεδίο `'q2_time'` περιέχει τον χρόνο που σημείωσε ο οδηγός στο δεύτερο μέρος των κατατακτηρίων δοκιμών (Q2).
- Το πεδίο `'q3_time'` περιέχει τον χρόνο που σημείωσε ο οδηγός στο τρίτο μέρος των κατατακτηρίων δοκιμών (Q3).
- Το πεδίο `'final_time'` περιέχει τον συνολικό χρόνο που χρειάστηκε ο οδηγός για να ολοκληρώσει τον αγώνα.
- Το πεδίο `'status'` περιγράφει την κατάσταση τερματισμού του οδηγού (π.χ. τερμάτισε, εγκατέλειψε, αποκλείστηκε).

- Το πεδίο `'points'` περιέχει τους βαθμούς που απονεμήθηκαν στον οδηγό για την επίδοσή του στη διαδικασία.

Πίνακας 4.10: Πίνακας με τα αποτελέσματα κάθε διαδικασίας (results) της Βάσης Δεδομένων

Name	Datatype	Length/Set
result_id	INT	11
session_id	INT	11
driver_id	INT	11
team_id	DATETIME	3
position	INT	3
classified_position	INT	11
grid_position	INT	11
q1_time	TIME	3
q2_time	TIME	3
q3_time	TIME	3
final_time	TIME	3
status	VARCHAR	50
points	DECIMAL	5,2

Στο Σχήμα 4.3 παρουσιάζεται το διάγραμμα οντοτήτων-συσχετίσεων (ER diagram) της Βάσης Δεδομένων της εφαρμογής, το οποίο δημιουργήθηκε με τη χρήση του λογισμικού dbdiagram.

Σχήμα 4.3: ER Diagram της Βάσης Δεδομένων

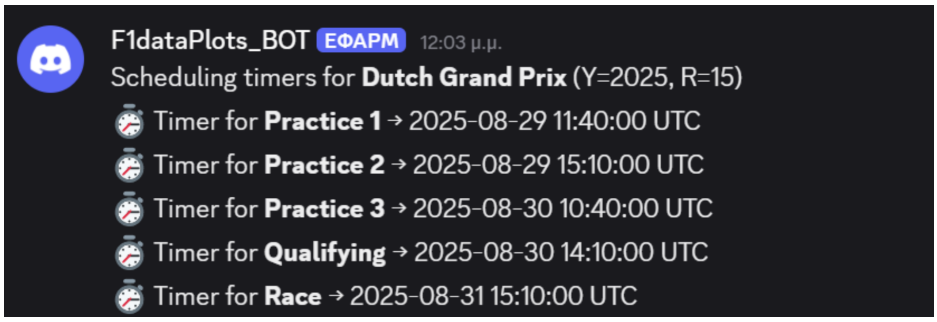
4.3.2 Δεδομένα από την βιβλιοθήκη FastF1

Για την άντληση των δεδομένων από την βιβλιοθήκη FastF1 δημιουργήθηκε ένα αυτοματοποιημένο σύστημα, το οποίο έχει ως σκοπό την άμεση καταχώρηση όλων των διαθέσιμων δεδομένων της F1 στη βάση δεδομένων αμέσως μετά την ολοκλήρωση κάθε διαδικασίας.

Το σύστημα υλοποιήθηκε με τη χρήση της ίδιας της βιβλιοθήκης FastF1 για την ανάκτηση του ημερολογίου των αγώνων, σε συνδυασμό με scripts που προγραμματίζουν επαναπροσθάπεια εκτέλεσης σε περίπτωση αποτυχίας και μηχανισμούς προγραμματισμού μέσω systemd σε επίπεδο χρήστη.

Ροή λειτουργίας του συστήματος

Η διαδικασίες του συστήματος ξεκινάνε κάθε Δευτέρα, όπου ένας εβδομαδιαίος systemd timer εκτελεί το script `plan_weekend.py`. Το script αυτό βρίσκει το προσεχές Grand Prix και δημιουργεί προσωρινούς (transient) timers για όλες της διαδικασίες του τριημέρου, όπως τις ελεύθερες δοκιμές, τις κατατακτήριες και τον αγώνα. Οι timers προγραμματίζονται να εκτελούνται λίγα λεπτά μετά τη λήξη κάθε διαδικασίας ώστε τα δεδομένα να είναι διαθέσιμα από την βιβλιοθήκη. Για την συγκεκριμένη διαδικασία όπως και για άλλες έχει δημιουργηθεί ένα κανάλι Discord για να παρακολουθείτε η ομαλή λειτουργία του συστήματος. Στην εικόνα 4.4 βλέπουμε την ειδοποίηση ότι δημιουργηθήκαν οι timers για το επόμενο GP.



Σχήμα 4.4: Ειδοποίηση ότι δημιουργηθήκαν οι timers για το επόμενο GP, από το plan_weekend.py

Για να διασφαλίσουμε ότι οι timers θα εκτελούνται ακόμη κι όταν ο χρήστης δεν είναι συνδεδεμένος, ενεργοποιήσαμε το "linger" για τον λογαριασμό ubuntu με την εντολή "`sudo loginctl enable-linger ubuntu`". Το "linger" (στο systemd) είναι μια ρύθμιση που επιτρέπει στον user-level systemd manager ενός χρήστη να μένει ενεργός χωρίς ο χρήστης να είναι συνδεδεμένος. Στη συνέχεια δημιουργήσαμε τον κατάλογο "`/.config/systemd/user`" και μέσα σε αυτόν προσθέσαμε τα δύο user units: το `plan-weekly.service` και το `plan-weekly.timer`.

Το timer `plan-weekly.timer` προγραμματίζει την εκτέλεση του service κάθε Δευτέρα (OnCalendar = Mon 10:00), ενώ με Persistent=true εξασφαλίζουμε ότι αν ο server είναι σβηστός την προγραμματισμένη ώρα, θα εκτελεστεί μόλις επανέλθει.

Κώδικας του plan-weekly.timer:

```
[Unit]
```

```
Description=Run plan_weekend.py every Monday to create next
 ↪  GP timers

[Timer]
OnCalendar=Mon 09:00
Persistent=true
RandomizedDelaySec=5min

[Install]
WantedBy=timers.target
```

Από την άλλη το `plan-weekly.service` είναι το user-level service που τρέχει μία φορά (Type = oneshot) το `plan_weekend.py`, από το virtualenv, όταν το ενεργοποιηθεί το timer.

Κώδικας του plan-weekly.service:

```
[Unit]
Description=Plan one-shot timers for the next F1 GP (uses
 ↪  plan_weekend.py)
After=network-online.target

[Service]
Type=oneshot
WorkingDirectory=/home/ubuntu/f1-bot
EnvironmentFile=/home/ubuntu/f1-bot/.env
Environment="PATH=/home/ubuntu/f1-bot/venv/bin:/usr/bin"
ExecStart=/home/ubuntu/f1-bot/venv/bin/python
 ↪  /home/ubuntu/f1-bot/plan_weekend.py --buffer-min 10
TimeoutStartSec=300
```

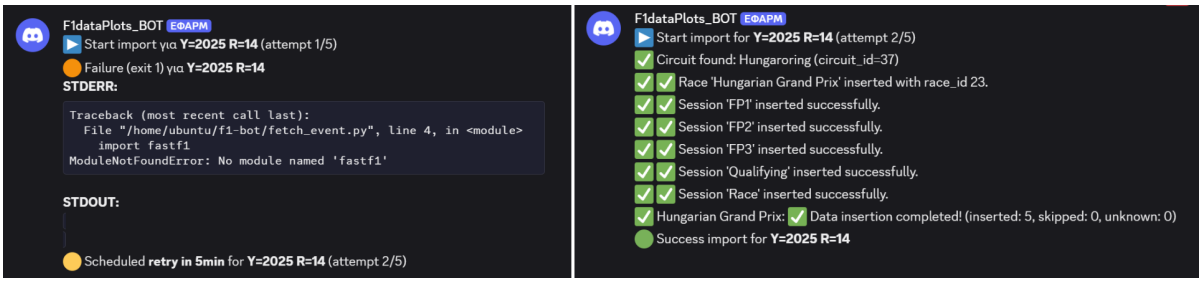
Στον παρακάτω κώδικα βλέπουμε πως γίνεται ο εντοπισμός του επόμενου GP από το ημερολόγιο αγώνων του FastF1, στο `plan_weekend.py`. Δίνουμε ως παράμετρο την τρέχουσα ώρα σε UTC (now_utc) και διαβάζουμε το πρόγραμμα του τρέχοντος έτους. Για κάθε γραμμή (event) που επιστρέφει παίρνουμε την ημερομηνία για την πρώτη διαθέσιμη διαδικασία (Session1DateUtc ή Session1Date), τη μετατρέπουμε σε ώρα UTC με το to_utc_aware(), και κρατάμε μόνο όσες είναι μετά το now_utc. Στη συνέχεια ταξινομούμε τις ημερομηνίες και επιλέγουμε την πιο κοντινή. Αν δεν βρεθεί κανένα event (π.χ. γιατί η σεζόν τελείωσε), επαναλαμβάνουμε το ίδιο για το επόμενο έτος.

Εύρεση επόμενου αγώνα από την βιβλιοθήκη FastF1:

```
def find_next_event(now_utc: datetime):
    year = now_utc.year

    def next_from_calendar(cal):
        cand = []
```

```
        for _, row in cal.iterrows():
            s1 = row.get("Session1DateUtc") or
             ↪  row.get("Session1Date")
            if not s1:
                continue
            s1u = to_utc_aware(s1)
            if s1u and s1u > now_utc:
                cand.append(row)
        if not cand:
            return None
        cand.sort(key=lambda r:
         ↪  to_utc_aware(r.get("Session1DateUtc") or
         ↪  r.get("Session1Date")))
        return cand[0]

    row = next_from_calendar(fastf1.get_event_schedule(year))
    if row is not None:
        return row

    return next_from_calendar(fastf1.get_event_schedule(year
     ↪  + 1))
```

Στην συνέχεια, όταν ενεργοποιηθεί ένας timer εκτελείται το `retry_runner.py` το οποίο λειτουργεί ως ενδιάμεσος μηχανισμός και έχει ως στόχο ότι η εφαρμογή python `f1-data-fetch` θα εκτελεστεί με τον σωστό τρόπο και όσες φορές χρειαστεί μέχρι να πετύχει. Έτσι η εκτέλεσή της πραγματοποιείτε χρησιμοποιώντας το σωστό περιβάλλον python του virtual environment και το σωστό working directory. Αν η εκτέλεση αποτύχει, ο wrapper (δηλαδή το αρχείο `retry_runner.py`) προγραμματίζει αυτόματα νέα προσπάθεια ανά πέντε λεπτά, μέχρι να φτάσει τον μέγιστο αριθμό επαναλήψεων, ενώ ενημερώνει σε κάθε στάδιο το Discord με σχετικά μηνύματα. Στην εικόνα 4.5 βλέπουμε στα αριστερά την αποτυχημένη εκτέλεση και στα δεξιά την επαναπροσπαθεια με επιτυχημένη εκτέλεση αυτή τη φορά.



Σχήμα 4.5: Ειδοποίηση για αποτυχημένη εκτέλεση (αριστερά) και ειδοποίηση για επιτυχημένη εκτέλεση (δεξιά)

Στον παρακάτω κώδικα βλέπουμε μέρος του `retry_runner.py`, όπου εκτελείτε η εφαρμογη `f1-data-fetch` με το Python του virtualenv, περνώντας το τρέχον περιβάλλον και ορίζοντας

το σωστό working directory, ενώ κρατα το standard output και standard error ως κείμενο. Αν η εκτέλεση επιστρέψει `returncode == 0`, στέλνει μήνυμα επιτυχίας στο Discord και τερματίζει με `exit(0)`. Αν αποτύχει, παίρνει τους τελευταίους 800 χαρακτήρες από το standard output και standard error (για σύντομο debugging) και στέλνει μήνυμα αποτυχίας, ενώ σε άλλο κώδικα θα προγραμματιστεί επαναπράσθαπεια εκτέλεσης. Τέλος, αν έχει ήδη φτάσει το όριο προσπαθειών (retry_count + 1 >= retry_max), ενημερώνει για οριστική αποτυχία και τελειώνει την εκτέλεση χωρίς άλλη επανάληψη.

Κώδικας εκτέλεσης του βασικού script, fetch_data.py:

```
# Running fetch
res = subprocess.run(
    [str(PYTHON_BIN), "-m", "app.main"],
    env=env,
    cwd=str(PROJECT_DIR),
    capture_output=True,
    text=True
)

if res.returncode == 0:
    notify(f"Success import for **Y={year} R={rnd}**")
    sys.exit(0)

# Send output for debugging.
out_tail = (res.stdout or "")[-800:]
err_tail = (res.stderr or "")[-800:]
notify(f"Failure (exit {res.returncode}) για **Y={year}
 ↪  R={rnd}**\n"
       f"**STDERR:**\n```\n{err_tail}\n```\n"
       f"**STDOUT:**\n```\n{out_tail}\n```")

# If the attempt limit is reached, stop fetch.
if retry_count + 1 >= retry_max:
    notify(f"Definitive failure for **Y={year} R={rnd}**
     ↪  after {retry_max} attempts")
    sys.exit(res.returncode)
```

Η εφαρμογή `f1-data-fetch` είναι μια python υλοποίηση που λειτουργεί σαν ένα πλήρες ETL pipeline (Extract (Εξαγωγή), Transform (Μετασχηματισμός) και Load (Φόρτωση)) για δεδομένα Formula 1. Ο ρόλος της είναι να αντλεί δεδομένα από τη βιβλιοθήκη FastF1, να τα καθαρίζει, να τα μετασχηματίζει σε κατάλληλη μορφή και τελικά να τα εισάγει στην βάση MySQL που έχουμε δημιουργήσει. Κατά τη διάρκεια της εκτέλεσης υπάρχει πρόβλεψη ώστε να μην γίνονται διπλοεγγραφές και παράλληλα να στέλνονται ειδοποιήσεις στο Discord ώστε να υπάρχει συνεχής εικόνα για την πορεία της διαδικασίας.

Για να είναι η υλοποίηση επεκτάσιμη και συντηρήσιμη, ο κώδικας χωρίστηκε σε διακριτά modules.

Υπάρχει το αρχείο με τις ρυθμίσεις που διαβάζει μεταβλητές από το περιβάλλον ή από το .env. Υπάρχει το αρχείο για τις ειδοποιήσεις στο Discord, το αρχείο για τη διασύνδεση με τη βάση μέσω SQLAlchemy, καθώς και τα βοηθητικά αρχεία για τις κλήσεις του API και για τη βασική λογική του pipeline. Τέλος, υπάρχει και το κεντρικό αρχείο, το main.py, που συνδέει όλα τα κομμάτια μεταξύ τους. Έτσι η εφαρμογή έχει καθαρή δομή και μπορεί να τρέχει είτε χειροκίνητα είτε αυτόματα μέσω systemd στον server.

Η εκτέλεση ξεκινά με τον έλεγχο των μεταβλητών. Αν λείπει κάποια βασική μεταβλητή, η εφαρμογή στέλνει μήνυμα σφάλματος και τερματίζει. Έπειτα δημιουργεί το connection προς τη βάση και κάνει έναν γρήγορο έλεγχο για να βεβαιωθεί ότι όλα λειτουργούν. Το connection δημιουργείται με SQLAlchemy και παραμένει διαθέσιμο σε όλο την εφαρμογή:

Κώδικας δημιουργίας σύνδεσης με την βάση MySql

```
def make_engine() -> Engine:
url = (
    f"mysql+mysqlconnector://{mysettings.DB_USER}:{mysetting⌋
     ↪  s.DB_PASS}@{mysettings.DB_HOST}/{mysettings.DB_NAME}"
)
engine = create_engine(url, pool_pre_ping=True,
 ↪  pool_recycle=300)
with engine.connect() as conn:
    conn.execute(text("SELECT 1"))
return engine
```

Στη συνέχεια γίνεται η ανάκτηση των στοιχείων του αγώνα με το `fastf1.get_event` και ακολουθεί η ενημέρωση των πινάκων `circuits` και `races`. Αν η πίστα ή ο αγώνας δεν υπάρχουν στη βάση, γίνεται η εισαγωγή τους και αμέσως μετά γίνεται η καταχώριση των διαθέσιμων διαδικασιών. Στον παρακάτω κώδικα φαίνεται η συνάρτηση που φροντίζει να εισάγει έναν αγώνα μόνο αν δεν υπάρχει ήδη. Αν υπάρχει, επιστρέφεται το id και στέλνεται ειδοποίηση στο Discord, αλλιώς γίνεται εισαγωγή και ενημερώνεται το χρήστης:

Κώδικας εισαγωγής ενός αγώνα στην βάση

```
name = event["EventName"]
date_str = event["EventDate"].strftime("%Y-%m-%d")

rid = conn.execute(
    text("SELECT race_id FROM races WHERE name = :n AND date
     ↪  = :d"),
    {"n": name, "d": date_str},
).scalar()

if rid:
    notify_ok(f"Race '{name}' already exists (id={rid}).")
    return rid
```

```
conn.execute(
    text(
        """INSERT INTO races (circuit_id, name, round,
        ↪  fullEventName, date, season_year)
        VALUES (:cid, :n, :r, :full, :d, :y)"""
    ),
    {
        "cid": circuit_id,
        "n": name,
        "r": int(event["RoundNumber"]),
        "full": event["OfficialEventName"],
        "d": date_str,
        "y": event["EventDate"].strftime("%Y"),
    },
)
```

Στον παρακάτω κώδικα βρίσκεται η επεξεργασία των διαδικασιών ενώς αγώνα. Επειδή τη στιγμή της εκτέλεσης μπορεί να είναι διαθέσιμες μόνο μερικές διαδικασίες, για παράδειγμα το FP1 και το FP2, ο κώδικας προσπαθεί να φορτώσει τα δεδομένα και προχωρά μόνο αν αυτά υπάρχουν. Αν δεν υπάρχουν, στέλνει μήνυμα ότι η διαδικασία (π.χ. FP2) παραλείπεται προσωρινά.

Κώδικας εκτέλεσης του βασικού script, fetch_data.py:

```
try:
        session_data = fastf1.get_session(year, round_number,
        ↪  session_name)
        session_data.load()
except Exception as e:
        notify_warn(f"Δε βρέθηκαν δεδομένα για
        ↪  '{session_type}' (ίσως δεν έχει διεξαχθεί ακόμη):
        ↪  {e}")
        not_available.append(session_type)
        session_number += 1
        continue
```

Με αυτό τον τρόπο η βάση ενημερώνεται σταδιακά, χωρίς να αποτυγχάνει ολόκληρη η εκτέλεση. Αν έχει διεξαχθεί μόνο το FP1, θα εισαχθεί κανονικά, ενώ τα υπόλοιπα θα σημειωθούν ως μη διαθέσιμα ακόμα.

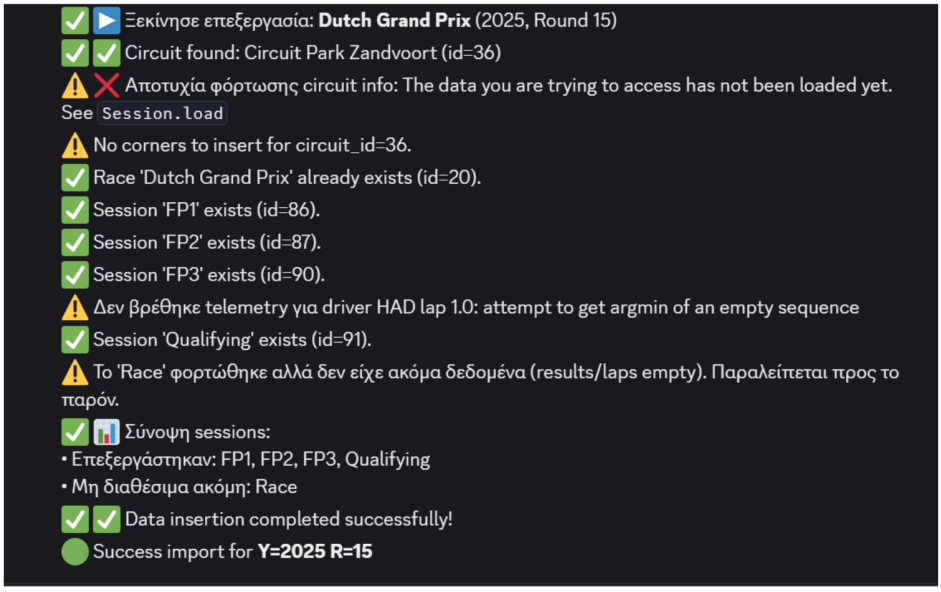
Κάθε βασικό βήμα συνοδεύεται από ένα try/except ώστε να μη σταματά η ροή και να υπάρχει συνεχής καταγραφή στο Discord. Το κομμάτι των ειδοποιήσεων έχει σπάσει σε τρεις μικρές συναρτήσεις, κάθε module μπορεί να στείλει μηνύματα για επιτυχία, προειδοποίηση ή σφάλμα.

Κώδικας για τις ενημερώσεις στο Discord

```
def _post_discord(message: str):
if not settings.DISCORD_WEBHOOK_URL:
```

```
    mysettings
try:
    requests.post(mysettings.DISCORD_WEBHOOK_URL,
     ↪  json={"content": message}, timeout=10)
except Exception:
    pass


def notify_ok(msg: str):
    _post_discord(f"Success {msg}")


def notify_warn(msg: str):
    _post_discord(f"Warning {msg}")


def notify_error(msg: str):
    _post_discord(f"Error {msg}")
```

Έτσι ο διαχειριστής μπορεί να βλέπει σε πραγματικό χρόνο αν εισήχθη ένα νέα πίστα, αν έγινε κάποια ενημέρωση για της διαδικασίες ή αν κάποια δεδομένα δεν ήταν ακόμη διαθέσιμα. Στο τέλος, αποστέλλεται μια σύνοψη για όσα επεξεργάστηκαν και όσα παραλείφθηκαν, ώστε να είναι ξεκάθαρη η εικόνα της βάσης σε κάθε χρονική στιγμή.



Σχήμα 4.6: Οι ειδοποιήσεις στο discord μετά από μια ολοκληρωμένη εκτέλεση της εφαρμογής

4.3.3 Web API

Το Web API αποτελεί ένα κρίσιμο τμήμα της εφαρμογής, καθώς μέσω αυτού πραγματοποιείται η επικοινωνία μεταξύ του Frontend και του Backend. Για να αποκτήσει κάποιος πρόσβαση στο Web API του «F1dataplots» και συνεπώς να κάνει χρήση των δεδομένων της εφαρμογής, απαιτείται η κλήση του κατάλληλου endpoint, έτσι ώστε να ανακτηθούν τα απαραίτητα δεδομένα. Κάθε endpoint, μετά την κατάλληλη κλήση, επιστρέφει το αποτέλεσμα σε μορφή JSON.

Πολλά endpoints απαιτούν συγκεκριμένες παραμέτρους (π.χ. `season`, `session_id`, κ.α.), ώστε να μπορούν να φιλτράρουν και να επιστρέψουν τα σωστά δεδομένα. Ο έλεγχος εγκυρότητας των παραμέτρων αυτών είναι απαραίτητος, καθώς αποτρέπει την εκτέλεση λανθασμένων ή ελλιπών αιτημάτων και διασφαλίζει ότι η απόκριση θα περιέχει σωστά δεδομένα. Για το σκοπό αυτό χρησιμοποιούνται κατάλληλοι μηχανισμοί χειρισμού σφαλμάτων και αποστολής αντίστοιχων μηνυμάτων προς τον χρήστη ή το σύστημα που καταναλώνει το API.

Το σύνολο των δημόσιων API endpoints της εφαρμογής φαίνεται αναλυτικά στον Πίνακα 4.11

Παρακάτω παρατίθενται μερικά παραδείγματα PHP κώδικα του Web API της εφαρμογής, προκειμένου να κατανοηθεί ο ρόλος του, που είναι η παροχή δεδομένων προς το χρήστη ή άλλα συστήματα. Μέσα από αυτά τα παραδείγματα, αναλύονται βασικές λειτουργίες που σχετίζονται με την ανάκτηση και το φιλτράρισμα των δεδομένων. Ειδικότερα, δίνονται τα παρακάτω τέσσερα παραδείγματα:

- Κώδικας αιτήματος προς την βάση για τις διαθέσιμες σεζον
- Κώδικας αιτήματος προς την βάση για τους διαθέσιμους αγώνες
- Κώδικας για την ανάκτηση των διαδικασιών ενώς αγώνα.
- Κώδικας για την ανάκτηση των δεδομένων τηλεμετρίας

Ο κώδικας που ακολουθεί αφορά το endpoint `seasons`, το οποίο εμφανίζει τις διαθέσιμες αγωνιστικές σεζόν της βάσης. Συγκεκριμένα, εκτελεί ερώτημα στη βάση δεδομένων για να επιστρέψει τις διακριτές χρονιές από τον πίνακα `driver_team_season`, ταξινομημένες κατά φθίνουσα σειρά και αποστέλλει την απόκριση σε μορφή JSON. Η λειτουργικότητα αυτή χρησιμοποιείται από το frontend για να γεμίσει επιλογές (dropdowns) και φίλτρα που απαιτούν λίστα με τις σεζόν.

Κώδικας PHP αιτήματος προς την βάση δεδομένων για τις διαθέσιμες σεζον:

```
\$query = "SELECT DISTINCT season_year FROM
 ↪  driver_team_season ORDER BY season_year DESC";
\$st = \$mysqli->prepare(\$query);
\$st->execute();
\$result = \$st->get_result();

\$seasons = [];
while (\$row = \$result->fetch_assoc()) {
    \$seasons[] = (int)\$row['season_year'];
```

Πίνακας 4.11: Public API Endpoints της Εφαρμογής

A/A	Μέθοδος HTTP	Endpoint	Λειτουργία
1	GET	/seasons	Επιστρέφει τις αγωνιστικές σεζόν που υπάρχουν διαθέσιμες στην βάση
2	GET	/races?season=2025	Για μια συγκεκριμένη σεζόν επιστρέφει μια λίστα με όλους τους αγώνες.
3	GET	/race-sessions? season=2025&round=1	Για μια συγκεκριμένη σεζόν και ένα round επιστρέφει τις διαδικασίες του αγώνα (FP1, FP2, FP3, Qualifying, Race)
4	GET	/session-drivers? season=2025&round=1& session=Qualifying	Για μια συγκεκριμένη διαδικασία ενός αγώνα επιστρέφει όλους τους οδηγούς που συμμετείχαν σε αυτή
5	GET	/laps? session_id=30&driver_id=8	Επιστρέφει τους γύρους για έναν οδηγό σε μια συγκεκριμένη διαδικασία (ή μόνο τον ταχύτερο με την παράμετρο fastest=true)
6	GET	/session-laps?session_id=30	Επιστρέφει τους γύρους για όλους τους οδηγούς σε μια συγκεκριμένη διαδικασία (ή μόνο τον ταχύτερο για τον κάθε οδηγό με την παράμετρο fastest=true)
7	GET	/telemetry?lap_id=191	Επιστρέφει τα δεδομένα τηλεμετρίας για τον ζητούμενο γύρο
8	GET	/telemetry?session_id=31& driver_id=7&lap_number=4	Επιστρέφει τα δεδομένα τηλεμετρίας για τον ζητούμενο γύρο
9	GET	/results?session_id=1	Επιστρέφει τα αποτελέσματα για μια διαδικασία
10	GET	/track_status?session_id=1	Επιστρέφει την κατάσταση της πίστας καθ΄ όλη την διάρκεια μιας διαδικασίας (όπως κίτρινη σημαία, κόκκινη σημαία κ.α.)
11	GET	/next-race-session	Επιστρέφει βασικές πληροφορίες για τον επόμενο αγώνα
12	GET	/last-session-info	Επιστρέφει βασικές πληροφορίες για την προηγούμενη διαδικασία που πραγματοποιήθηκε
13	GET	/pit-stop?session_id=5	Επιστρέφει τις αλλαγές ελαστικών που πραγματοποιήθηκαν κατά την διάρκεια ενός αγώνα
14	GET	/drivers-standings?season=2025	Επιστρέφει τις βαθμολογίες των οδηγών για την ζητούμενη σεζόν
14	GET	/teams-standings?season=2025	Επιστρέφει τις βαθμολογίες των ομάδων για την ζητούμενη σεζόν

}

```
        header('Content-type: application/json');
        echo json_encode(["seasons" => \$seasons],
         ↪  JSON_PRETTY_PRINT);
```

Ο κώδικας που ακολουθεί πραγματοποιεί την ανάκτηση όλων των αγώνων μίας συγκεκριμένης σεζόν. Ελέγχει αν παρέχεται η παράμετρος `season` και αν είναι αριθμητική, αν δεν ικανοποιούνται τα κριτήρια επιστρέφει το σχετικό μήνυμα σφάλματος, με κατάσταση HTTP 400. Εάν η παράμετρος είναι εντάξει τότε εκτελεί το ερώτημα που επιστρέφει βασικά στοιχεία αγώνων, πληροφορίες της πίστας και ένδειξη αν υπάρχει αγώνας sprint και στέλνει την απόκριση σε μορφή JSON.

Κώδικας PHP αιτήματος προς την βάση για τους διαθέσιμους αγώνες:

```
$season = $params['season'] ?? null;

if (!$season || !is_numeric($season)) {
    header('HTTP/1.1 400 Bad Request');
    echo json_encode(["error" => "Season parameter is required
    and must be a number"]);
    exit;
}

$query = "SELECT r.race_id, r.name, r.round, r.date, c.name AS
        EXISTS (
            SELECT 1 FROM sessions s
            WHERE s.race_id = r.race_id AND
            s.session_type = 'Sprint') AS sprint
        FROM races r
        JOIN circuits c ON r.circuit_id = c.circuit_id
        WHERE YEAR(r.date) = ?
        ORDER BY r.date ASC";
$st = $mysqli->prepare($query);
$st->bind_param('i', $season);
$st->execute();
$result = $st->get_result();

$races = [];
while ($row = $result->fetch_assoc()) {
    $races[] = [
        "race_id" => (int)$row['race_id'],
        "name" => $row['name'],
        "round" => $row['round'],
        "date" => $row['date'],
        "country" => $row['country'],
        "location" => $row['location'],
        "circuitname" => $row['circuit_name'],
```

```
        "sprint" => (bool)$row['sprint']
    ];
}

header('Content-type: application/json');
echo json_encode(["season" => (int)\$season, "races" => \$races],
    JSON_PRETTY_PRINT);
```

Ο παρακάτω κώδικας πραγματοποιεί την ανάκτηση όλων των διαθέσιμων διαδικασιών (ελεύθερες δοκιμές, κατατακτήριες, αγώνας) που σχετίζονται με έναν αγώνα μίας συγκεκριμένης σεζόν. Για την κλήση του απαιτείται η παράμετρος `season`, η οποία πρέπει να είναι αριθμητική, καθώς και μία επιπλέον παράμετρος για την αναγνώριση του αγώνα: είτε ο αριθμός γύρου (`round`), είτε το όνομα του αγώνα (`race_name`).

Σε περίπτωση που οι παρεχόμενες παράμετροι δεν είναι έγκυρες ή αν δεν βρεθούν δεδομένα για τις τιμές τους, το API επιστρέφει κατάλληλα μηνύματα σφάλματος με τους αντίστοιχους HTTP κωδικούς. Εάν οι παράμετροι είναι εντάξει, τότε επιστρέφει τις διαδικασίες που αντιστοιχούν στον ζητούμενο αγώνα. Τα δεδομένα που επιστρέφονται περιλαμβάνουν τον μοναδικό κωδικό της διαδικασίας, τον τύπο της διαδικασίας (π.χ. ελεύθερες δοκιμές, κατατακτήριες, αγώνας) και την ώρα έναρξής της.

Κώδικας PHP για την ανάκτηση των διαδικασιών ενώς αγώνα:

```
    $season = $params['season'] ?? null;
    $round = $params['round'] ?? null;
    $race_name = $params['race_name'] ?? null;

    if (!$season || !is_numeric($season)) {
        header('HTTP/1.1 400 Bad Request');
        echo json_encode(["error" =>
            "Season parameter is required and must be a number"]);
        exit;
    }

    if (!$round && !$race_name) {
        header('HTTP/1.1 400 Bad Request');
        echo json_encode(["error" =>
            "Either round or race name parameter is required"]);
        exit;
    }

    if ($round && !is_numeric($round)) {
        header('HTTP/1.1 400 Bad Request');
        echo json_encode(["error" =>
            "Round must be a number if provided"]);
```

```
    exit;
}

if ($round && is_numeric($round)) {
    $query = "SELECT r.race_id, r.name,
                     s.session_id, s.session_type,
                     s.start_time
              FROM races r
              JOIN sessions s ON r.race_id = s.race_id
              WHERE r.season_year = ? AND r.round = ?
              ORDER BY s.start_time ASC";
    $st = $mysqli->prepare($query);
    $st->bind_param('ii', $season, $round);
} else {
    $query = "SELECT r.race_id, r.name,
                     s.session_id, s.session_type,
                     s.start_time
              FROM races r
              JOIN sessions s ON r.race_id = s.race_id
              WHERE r.season_year = ? AND r.name = ?
              ORDER BY s.start_time ASC";
    $st = $mysqli->prepare($query);
    $st->bind_param('is', $season, $race_name);
}

$st->execute();
$result = $st->get_result();

$sessions = [];
while ($row = $result->fetch_assoc()) {
    $sessions[] = [
        "session_id" => $row['session_id'],
        "session_type" => $row['session_type'],
        "start_time" => $row['start_time']
    ];
    $race_name_result = $row['name'];
    $race_id_result = (int)$row['race_id'];
}

if (empty($sessions)) {
    header('HTTP/1.1 404 Not Found');
    echo json_encode(["error" =>
        "No sessions found for given parameters"]);
    exit;
}
```

```
header('Content-type: application/json');
echo json_encode([
    "season" => (int)$season,
    "race_id" => $race_id_result,
    "race_name" => $race_name_result,
    "sessions" => $sessions
], JSON_PRETTY_PRINT);
```

Τέλος ο παρακάτω κώδικας πραγματοποιεί την ανάκτηση όλων των διαθέσιμων δεδομένων τηλεμετρίας που αντιστοιχούν σε έναν συγκεκριμένο γύρο. Μέσα από την κλήση του endpoint, ο χρήστης μπορεί να αντλήσει πληροφορίες όπως η ταχύτητα του μονοθεσίου, το ποσοστό χρήσης γκαζιού (*throttle*), η χρήση του φρένου (*brake*), οι στροφές του κινητήρα ανά λεπτό (*RPM*), η σχέση του κιβωτίου ταχυτήτων, καθώς και η ενεργοποίηση του *DRS*. Επιπλέον, επιστρέφονται τα χρονικά σημεία καταγραφής (*timestamps*), η απόσταση που έχει διανυθεί, ο χρόνος του session και ο επιμέρους χρόνος γύρου.

Κώδικας PHP για την ανάκτηση των δεδομένων τηλεμετρίας:

```
$lap_id = $params['lap_id'] ?? null;
$session_id = $params['session_id'] ?? null;
$driver_id = $params['driver_id'] ?? null;
$lap_number = $params['lap_number'] ?? null;

if ($lap_id && is_numeric($lap_id)) {
    $query = "SELECT timestamp, rpm, speed,
                n_gear, throttle,
                brake, drs, distance,
                session_time, time
              FROM telemetry
              WHERE lap_id = ?
              ORDER BY timestamp ASC";
    $st = $mysqli->prepare($query);
    $st->bind_param('i', $lap_id);
    $st->execute();
    $result = $st->get_result();

} elseif ($session_id && $driver_id && $lap_number &&
is_numeric($session_id) && is_numeric($driver_id) &&
is_numeric($lap_number)) {
    $query = "SELECT t.timestamp, t.rpm, t.speed,
                t.n_gear, t.throttle,
                t.brake, t.drs, t.distance,
                t.session_time, t.time
```

```
                FROM telemetry t
                JOIN laps l ON t.lap_id = l.lap_id
                WHERE l.session_id = ?
                  AND l.driver_id = ?
                  AND l.lap_number = ?
                ORDER BY t.timestamp ASC";
        $st = $mysqli->prepare($query);
        $st->bind_param('iii', $session_id,
            $driver_id, $lap_number);
        $st->execute();
        $result = $st->get_result();

    } else {
        header('HTTP/1.1 400 Bad Request');
        echo json_encode(["error" =>
            "Required parameters: lap_id OR
            (session_id, driver_id AND lap_number)"]);
        exit;
    }
    ...
```

4.4 Υλοποίηση του Frontend

Ένα εξίσου σημαντικό μέρος του «F1dataplots» είναι το Frontend, το οποίο επιτρέπει την αξιοποίηση όλων των λειτουργιών που παρέχει η εφαρμογή μέσω μίας σύγχρονης και εύχρηστης γραφικής διεπαφής. Η επικοινωνία με το Web API πραγματοποιείται μέσω αιτημάτων HTTP, τα οποία επιστρέφουν δεδομένα σε μορφή JSON, τα οποία στη συνέχεια προβάλλονται με φιλικό τρόπο στον χρήστη, ώστε να μπορεί να δημιουργεί τα γραφήματα που επιθυμεί για την ανάλυση του.

Για την ανάπτυξη του Frontend αξιοποιείται το React, σε συνδυασμό με TypeScript και σύνταξη JSX, που επιτρέπουν την ανάπτυξη της εφαρμογής με στόχο την επαναχρησιμοποίηση κώδικα. Η αρχιτεκτονική του Frontend βασίζεται σε τέσσερα κύρια μέρη: το κεντρικό αρχείο App.tsx, το οποίο αποτελεί τον βασικό δρομολογητή της εφαρμογής, τα components, που περιέχουν επαναχρησιμοποιήσιμα κομμάτια της διεπαφής, τον φάκελο με τις σελίδες (pages) και τα αρχεία css που καθορίζουν την εμφάνιση των components και των σελίδων. Στις ενότητες που ακολουθούν παρουσιάζονται αναλυτικά τα παραπάνω στοιχεία, με παραδείγματα κώδικα και επεξήγηση της λειτουργικότητάς τους.

4.4.1 Η κεντρική δομή της εφαρμογής — το αρχείο App.tsx

Το αρχείο `App.tsx` αποτελεί το κεντρικό σημείο εκκίνησης του Frontend της εφαρμογής και ουσιαστικά είναι υπεύθυνο για την πλοήγηση μεταξύ των σελίδων. Σε αυτό το αρχείο αρχικοποιείται

ο `BrowserRouter` από τη βιβλιοθήκη `React Router`, μέσω του οποίου καθορίζονται όλες οι πιθανές διαδρομές (routes) που μπορεί να ακολουθήσει ο χρήστης μέσα στην εφαρμογή.

Η χρησιμότητα του `App.tsx` έγκειται στο ότι συγκεντρώνει και οργανώνει τις κύριες σελίδες (pages) και τα αντίστοιχα components της εφαρμογής, αντιστοιχίζοντας κάθε URL σε μία συγκεκριμένη λειτουργικότητα. Με αυτόν τον τρόπο επιτυγχάνεται η υλοποίηση της εφαρμογής ως Single Page Application (SPA), όπου ο χρήστης μπορεί να περιηγείται σε διαφορετικές ενότητες χωρίς να απαιτείται ανανέωση της σελίδας από τον φυλλομετρητή.

Απόσπασμα κώδικα δρομολόγησης React (App.tsx)

```
function ChartsRouter() {
    const { chartName } = useParams();

    switch (chartName) {
        case "max-speed":
          return <MaxSpeedChart />;
        case "laps-distribution":
          return <LapsDistributionChart />;
        case "reaction-time":
          return <ReactionTimeChart />;
        default:
          return <div>Chart not found</div>;
    }
}

function App() {
    return (
       <BrowserRouter basename="/">
         <Routes>
          <Route path="/" element={<HomePage />} />
           <Route path="
               /race-analysis/:raceSlug/:sessionType/:chartName"
             element={<Layout><ChartsRouter /></Layout>} />
         </Routes>
       </BrowserRouter>
    );
}
```

Συνολικά, το αρχείο `App.tsx` είναι απαραίτητο γιατί συντονίζει όλη την πλοήγηση της εφαρμογής και επιτρέπει την εύκολη προσθήκη νέων σελίδων ή λειτουργιών, απλά με την προσθήκη νέων διαδρομών.

4.4.2 Τα components στο React

Τα components αποτελούν το βασικό δομικό στοιχείο κάθε React εφαρμογής, καθώς κάθε component αποτελεί μια ανεξάρτητη και επαναχρησιμοποιήσιμη μονάδα τόσο σε λογική όσο και σε εμφάνιση. Μέσα από τα component, η εφαρμογή οργανώνεται σε μικρότερα και ανεξάρτητα κομμάτια κώδικα, τα οποία μπορούν εύκολα να συνδυαστούν ώστε να σχηματίσουν ολόκληρες σελίδες ή σύνθετες διεπαφές χρήστη.

Η χρησιμότητα των components είναι ιδιαίτερα σημαντική, καθώς συμβάλλουν στη βελτίωση της επαναχρησιμοποίησης κώδικα, αφού το ίδιο component μπορεί να ενσωματωθεί σε πολλά διαφορετικά σημεία της εφαρμογής χωρίς να χρειάζεται να ξαναγραφεί ο κώδικας. Παράλληλα, κάθε component αναλαμβάνει μία συγκεκριμένη λειτουργία της εφαρμογής κάνοντας τον κώδικα πιο καθαρό και κατανοητό. Τέλος, η modular αρχιτεκτονική τους διευκολύνει τη συντήρηση και την επεκτασιμότητα της εφαρμογής, καθώς καθιστά πιο απλή την τροποποίηση ή την προσθήκη νέων λειτουργιών.

Στην εφαρμογή «F1dataplots» τα components αξιοποιούνται για την απεικόνιση δεδομένων, την εμφάνιση πινάκων και την υλοποίηση στοιχείων διεπαφής (όπως κουμπιά, μενού επιλογών κ.α.). Με τον τρόπο αυτό, το frontend αποκτά έναν ευέλικτο χαρακτήρα, προσαρμόζοντας την εμφάνιση και τη λειτουργικότητά του ανάλογα με τις απαιτήσεις.

Παράδειγμα Component: RaceCard

Ένα από τα χαρακτηριστικά components της εφαρμογής είναι το `RaceCard`. Το συγκεκριμένο component είναι υπεύθυνο για την απεικόνιση των βασικών πληροφοριών ενός αγώνα (όνομα, χώρα, πόλη, ημερομηνία, πίστα κ.ά.). Παράλληλα, δίνει τη δυνατότητα μετάβασης στη σελίδα ανάλυσης του αγώνα (`/race-analysis`) όταν ο χρήστης επιλέξει την κάρτα.

Με αυτόν τον τρόπο, το `RaceCard` λειτουργεί τόσο ως γραφικό στοιχείο παρουσίασης δεδομένων όσο και ως μέσο αλληλεπίδρασης, διευκολύνοντας τον χρήστη να μεταβεί στην σελίδα ενός συγκεκριμένου Grand Prix. Παρακάτω φαίνεται ο κώδικας του component και πως στην συνέχεια χρησιμοποιείτε μέσα σε μια σελίδα.

Παράδειγμα Κώδικα — Component RaceCard:

```
const RaceCard: React.FC<RaceCardProps> = ({
  raceId,
  round,
  name,
  country,
  city,
  year,
  dateRange,
  sprint,
  circuitname,
  completed
```

```
}) => {
  const navigate = useNavigate();

  const handleClick = () => {
    const slug =
      `${name.toLowerCase()
        .replace(/\s+/g, '-')
        .replace(/[^\w-]/g, '')}-${year}`;
    navigate(`/race-analysis/
        ${encodeURIComponent(slug)}?raceId=${raceId}`);
  };

  const flagSrc = `/flags/${country.toLowerCase()
    .replace(/\s+/g, "-")
    .replace(/[^\w-]/g, "")}-flag.png`;

  const trackName = (circuitname ?? "")
    .replace(/\s+/g, '-')
    .replace(/[^a-zA-Z0-9-]/g, '');
  const trackSrc = `/tracks/${trackName}.png`;

  return (
    <div className={styles.raceCard} onClick={handleClick}>
      <div className={styles.roundLabel}>
        <span className={styles.roundWord}>ROUND {round}</span>
        <div className={styles.raceFlag}>
          <img src={flagSrc} alt={`${country} flag`} />
        </div>
      </div>

      <div className={styles.raceName}>
        <span>{name.toUpperCase()}</span>
      </div>

      {city && <div className={styles.raceCity}>{city}</div>}
      <div className={styles.raceDate}>{dateRange}</div>
      {sprint && <div
        className={styles.sprintFormat}>*Sprint Format
        </div>}
      {completed && <div
        className={styles.completed}>COMPLETED
        </div>}

      <div className={styles.racetrack}>
        <img src={trackSrc} alt={circuitname} />
```

```
        </div>
      </div>
    );
  };

  export default RaceCard;
```

Η χρήση του component `RaceCard` σε μία σελίδα επιτρέπει την εμφάνιση όλων των διαθέσιμων αγώνων με οργανωμένο και ομοιόμορφο τρόπο. Το κάθε αντικείμενο του πίνακα `races` μετατρέπεται σε μία ξεχωριστή κάρτα αγώνα:

Παράδειγμα Κώδικα — Χρήση του RaceCard σε μια σελίδα:

```
<div className={styles.grid}>
    {races.map((race) => {
        const year = race.date.slice(0, 4);
        return (
          <RaceCard
            key={race.race_id}
            raceId={race.race_id}
            round={race.round}
            name={race.name}
            country={race.country}
            city={race.location}
            year={year}
            dateRange={getRaceDatesRange(race.date)}
            sprint={race.sprint}
            circuitname={race.circuitname}
            completed={isRaceCompleted(race.date)}
          />
        );
    })}
</div>
```

Παράδειγμα Component: ToolsButtons

Ένα ακόμη χαρακτηριστικό component που χρησιμοποιείται στην εφαρμογή είναι το `ToolsButtons`. Σκοπός του είναι να συγκεντρώνει και να παρέχει στον χρήστη τα βασικά εργαλεία που χρειάζεται κάθε γράφημα, όπως η δυνατότητα εξαγωγής του σε εικόνα (PNG), η λήψη των δεδομένων σε μορφή CSV και η προβολή επιπρόσθετων πληροφοριών (Info).

Το component είναι πλήρως παραμετροποιήσιμο, καθώς τα διαθέσιμα κουμπιά ορίζονται μέσα από έναν πίνακα ιδιοτήτων. Η εμφάνιση τους εξαρτάται από την παράμετρο `visibleButtons`, ενώ η λειτουργικότητά τους καθορίζεται μέσω συναρτήσεων που δίνονται ως props (`onExportPNG`, `onExportCSV`, `onInfo`). Με αυτόν τον τρόπο, το ίδιο component μπορεί να ενσωματωθεί σε οποιοδήποτε γράφημα, προσφέροντας μια συνεχή εμπειρία χρήσης.

Παράδειγμα Κώδικα — Component ToolsButtons:

```
export const ToolsButtons: React.FC<ToolsButtonsProps> = ({
    onExportPNG,
    onExportCSV,
    onInfo,
    visibleButtons,
}) => {
    const handlers = { onExportPNG, onExportCSV, onInfo };
    const [hovered, setHovered] = useState<string | null>(null);

    const buttonsToShow = BUTTONS.filter(
        btn => !visibleButtons || visibleButtons.includes(btn.id)
    );

    return (
        <div className={styles.toolsButtonsRow}>
        {buttonsToShow.map((btn) => (
           <div
              key={btn.id}
              className={styles.iconBtnWrapper}
              onMouseEnter={() => setHovered(btn.id)}
              onMouseLeave={() => setHovered(null)}
            >
            <button
                type="button"
                className={styles.iconBtn}
                onClick={
                    handlers[btn.onClickProp as keyof typeof handlers]
                }
                aria-label={btn.label}
            >
                <img src={btn.icon} alt={btn.label}
                    className={styles.iconImg} />
            </button>
            {hovered === btn.id && (
               <span className={styles.tooltip}>{btn.label}</span>
             )}
            </div>
        ))}
        </div>
    );
};
```

Η ενσωμάτωση του `ToolsButtons` σε ένα γράφημα γίνεται με ιδιαίτερη ευκολία, απλώς με την προσθήκη του component και τη σύνδεσή του με τις κατάλληλες συναρτήσεις που χειρίζονται την εξαγωγή δεδομένων ή εικόνας, καθώς και την εμφάνιση πληροφοριών:

Παράδειγμα Κώδικα — Χρήση του ToolsButtons σε ένα γράφημα:

```
<ToolsButtons
    onExportPNG={handleExportPNG}
    onExportCSV={handleExportCSV}
    onInfo={() => setShowInfo(true)}
/>
{showInfo && (
    <InfoModal
       onClose={() => setShowInfo(false)}
       infoText="Best sector times ανά οδηγό."
    />
)}
```

Το μόνο που χρειάστηκε να προστεθεί επιπλέον ήταν οι συναρτήσεις που διαχειρίζονται τον τρόπο με τον οποίο θα πραγματοποιηθεί η εξαγωγή των δεδομένων ή της εικόνας. Με αυτόν τον τρόπο, το component `ToolsButtons` απλοποιεί σημαντικά την ανάπτυξη της εφαρμογής, αξιοποιώντας το component σε όλα τα γραφήματα.

4.4.3 Ο φάκελος με τις σελίδες της εφαρμογής (pages)

Ο φάκελος `pages` περιέχει, όπως υποδηλώνει και το όνομά του, όλες τις κύριες σελίδες της εφαρμογής «F1dataplots». Κάθε αρχείο μέσα σε αυτόν αντιστοιχεί σε μια σελίδα, η οποία συνδυάζει επιμέρους components, προκειμένου να παρέχει στον χρήστη οργανωμένα όλες τις λειτουργίες και τις δυνατότητες της εφαρμογής. Έτσι, κάθε σελίδα εξυπηρετεί έναν συγκεκριμένο σκοπό.

Στον φάκελο αυτό περιλαμβάνεται και ο υποφάκελος `charts`, όπου βρίσκονται τα αρχεία που υλοποιούν τα γραφήματα της εφαρμογής (π.χ. `MaxSpeedChart.tsx`, `LongRunsDistribution.tsx`). Σε καθένα από αυτά τα αρχεία περιέχονται οι κώδικες που δημιουργούν το αντίστοιχο γράφημα, αντλώντας δεδομένα από το Web API.

Επιπλέον, τα αρχεία `Home.tsx`, `Dashboard.tsx` και `ApiDocs.tsx` υλοποιούν τις βασικές σελίδες της εφαρμογής. Το `Dashboard` λειτουργεί ως κύρια οθόνη διαχείρισης, το `ApiDocs` παρέχει αναλυτικές πληροφορίες για το Web API, ενώ το `Home` αποτελεί την αρχική σελίδα της εφαρμογής με τις πρώτες οδηγίες προς τον χρήστη.

4.4.4 Ο φάκελος με τα αρχεία μορφοποίησης (css)

Ο φάκελος `css` περιλαμβάνει τα αρχεία που αφορούν τη μορφοποίηση και την οπτική εμφάνιση της εφαρμογής. Σε αυτόν βρίσκονται τόσο τα αρχεία που σχετίζονται με τα επιμέρους components

όσο και εκείνα που αφορούν τις κύριες σελίδες (pages). Κάθε αρχείο CSS είναι υπεύθυνο για την εμφάνιση συγκεκριμένων στοιχείων, εξασφαλίζοντας ότι η διεπαφή ακολουθεί μια ομοιόμορφη δομή και ταυτόχρονα παραμένει λειτουργική και ευχάριστη για τον χρήστη.

Η χρήση ξεχωριστών αρχείων μορφοποίησης επιτρέπει τον σαφή διαχωρισμό ανάμεσα στη λογική υλοποίησης (με React και TypeScript) και στην εμφάνιση (μέσω CSS). Με αυτόν τον τρόπο, επιτυγχάνεται μεγαλύτερη ευελιξία στη συντήρηση του κώδικα, καθώς τυχόν αλλαγές στο στυλ μπορούν να πραγματοποιηθούν χωρίς να επηρεαστεί η λειτουργικότητα του κώδικα.

4.4.5 Δημιουργία γραφημάτων στο Frontend

Για την υλοποίηση των γραφημάτων στην εφαρμογή χρησιμοποιήθηκε η βιβλιοθήκη **Chart.js** σε συνδυασμό με το **react-chartjs-2** και η βιβλιοθήκη **Plotly.js** συνδυασμό με το **react-plotly-2**.

Παρακάτω παρουσιάζονται οι βασικές διαδικασίες που ακολουθούνται κατά την δημιουργία όλων των γραφημάτων της εφαρμογής. Ανεξάρτητα από το είδος του γραφήματος η ανάπτυξη του κώδικα βασίζεται σε κοινά βήματα που παρουσιάζονται παρακάτω, μέσω ενός παραδείγματος.

Παράδειγμα Γραφήματος: MaxSpeedChart

Στο παρακάτω παράδειγμα παρουσιάζεται το component `MaxSpeedChart`, το οποίο εμφανίζει τις μέγιστες ταχύτητες που κατέγραψαν οι οδηγοί κατά τον ταχύτερο γύρο τους.

Βασικά στάδια δημιουργίας ενώς γραφήματος:

1. **Ανάκτηση δεδομένων από το API**

   Κατά τη φόρτωση της σελίδας, γίνεται το κατάλληλο αίτημα στο API, ώστε να ληφθούν οι τιμές της μέγιστης ταχύτητας για κάθε οδηγό. Τα δεδομένα αποθηκεύονται στην κατάσταση (`state`) του React component:

```
useEffect(() => {
    fetch(`/api/telemetry?
        data=max-speed&fastest-
        lap=true&session_id=${sessionId}`)
    .then((res) => res.json())
    .then((json) => {
        const speeds: SpeedData[] =
        json.fastest_laps_speeds_max_speed || [];
        speeds.sort((a, b) => b.max_speed - a.max_speed);
        setData(speeds);
      })
    .catch((err) => setError(err.message));
}, [sessionId]);
```

2. **Επεξεργασία των δεδομένων για το γράφημα**

   Τα δεδομένα που επιστρέφει το API επεξεργάζονται και οργανώνονται ώστε να χρησιμοποιηθούν από τη βιβλιοθήκη Chart.js. Χρησιμοποιούνται ετικέτες (labels) για οδηγούς ή ομάδες, τιμές (max speed) και τα χρώματα που αντιστοιχούν σε κάθε ομάδα.

```
const chartData = {
   labels: data.map((d) => d.driver_abbr),
   datasets: [
        {
          label: "Max Speed (km/h)",
          data: data.map((d) => d.max_speed),
          backgroundColor: data.map((d) =>
            teamColors[d.team] || "gray"),
        },
    ],
};
```

3. **Παραμετροποίηση του γραφήματος**

   Μέσω των `chartOptions` ορίζονται όλες οι απαραίτητες ρυθμίσεις για το γράφημα (όπως οι άξονες, οι τίτλοι, τα tooltips, τα animations κ.α.), ώστε να είναι κατανοητό στον χρήστη, να παρουσιάζει την κατάλληλη διαδραστικότητα και να προσαρμόζεται στις εκάστοτε απαιτήσεις οπτικοποίησης των δεδομένων.

```
const chartOptions: any = {
  responsive: true,
  plugins: {
    tooltip: {
      callbacks: {
        label: function (context: any) {
          const idx = context.dataIndex;
          if (mode === "drivers") {
            const speed = context.parsed.y;
            const driver = chartLabels[idx];
            const team = data.find((d) =>
                d.driver_abbr === driver)?.team || "";
            return `${driver} (${team}): ${speed} km/h`;
          } else {
            const speed = context.parsed.y;
            const team = chartLabels[idx];
            const drivers = chartDrivers[idx].join(", ");
            return `${team} (Driver: ${drivers}):
                ${speed} km/h`;
          }
```

```
          },
        },
      },
      datalabels: {
        anchor: "end",
        align: "end",
        formatter: (value: number) => value.toFixed(0),
        color: "black",
        display: true,
      },
      ...
    },
};
```

4. **Απεικόνιση του γραφήματος**

   Το τελικό γράφημα αποδίδεται μέσω του component `Bar` της βιβλιοθήκης **react-chartjs-2**, όπου περνάμε τα δεδομένα και τις επιλογές μορφοποίησης:

```
<Bar ref={chartRef} data={chartData}
    options={chartOptions} />
```

5. **Εργαλεία του γραφήματος**

   Το κάθε γράφημα συνοδεύεται από διαφορετικά εργαλεία, ανάλογα με τον τύπο και τον σκοπό του. Στην περίπτωση του `MaxSpeedChart`, εκτός από τα γενικά εργαλεία εξαγωγής (Export PNG, Export CSV, Info), υπάρχει και η δυνατότητα επιλογής οδηγών ή ομάδων μέσω ενός `CustomMultiSelect`. Το component αυτό επιτρέπει στον χρήστη να φιλτράρει τα δεδομένα του γραφήματος, επιλέγοντας μόνο τα στοιχεία που τον ενδιαφέρουν.

```
<div className={styles.selectRow}>
    <CustomMultiSelect
       options={mode === "drivers" ? driverOptions :
           teamOptions}
       value={mode === "drivers" ? selectedDrivers :
            selectedTeams}
       onChange={mode === "drivers" ? setSelectedDrivers :
            setSelectedTeams}
       icon={<span role="img" aria-label="palette"></span>}
       placeholder={mode === "drivers" ? "Select drivers" :
            "Select teams"}
       selectAllLabel="Select All"
    />
</div>
```

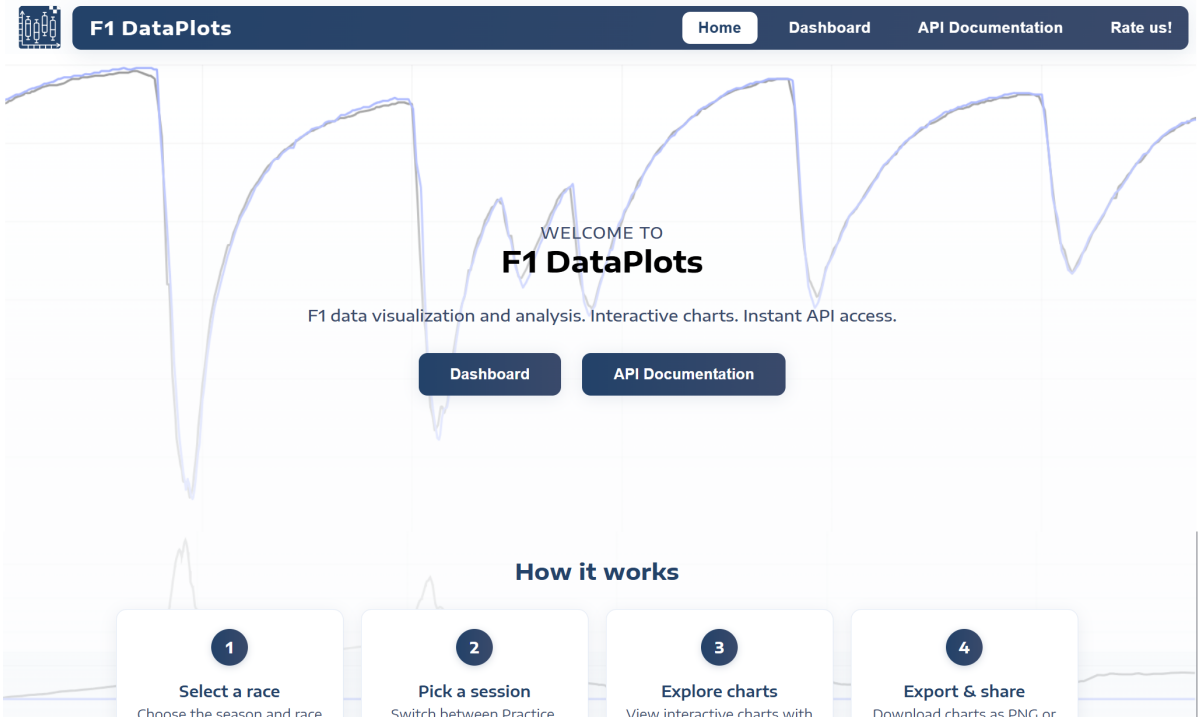
Συνολικά, κάθε γράφημα περιέχει όλα τα παραπάνω βασικά στοιχεία. Ωστόσο, όπως αναφέρθηκε και νωρίτερα, κάθε γράφημα έχει και τις δικές του ιδιαίτερες απαιτήσεις, τόσο ως προς τα δεδομένα που επεξεργάζεται όσο και ως προς τα εργαλεία που προσφέρει στον χρήστη. Έτσι, η εφαρμογή παραμένει ευέλικτη και προσαρμόζεται στο εκάστοτε σενάριο ανάλυσης, εξασφαλίζοντας ότι κάθε οπτικοποίηση εξυπηρετεί με τον καλύτερο δυνατό τρόπο τις ανάγκες του χρήστη.

Κεφάλαιο 5

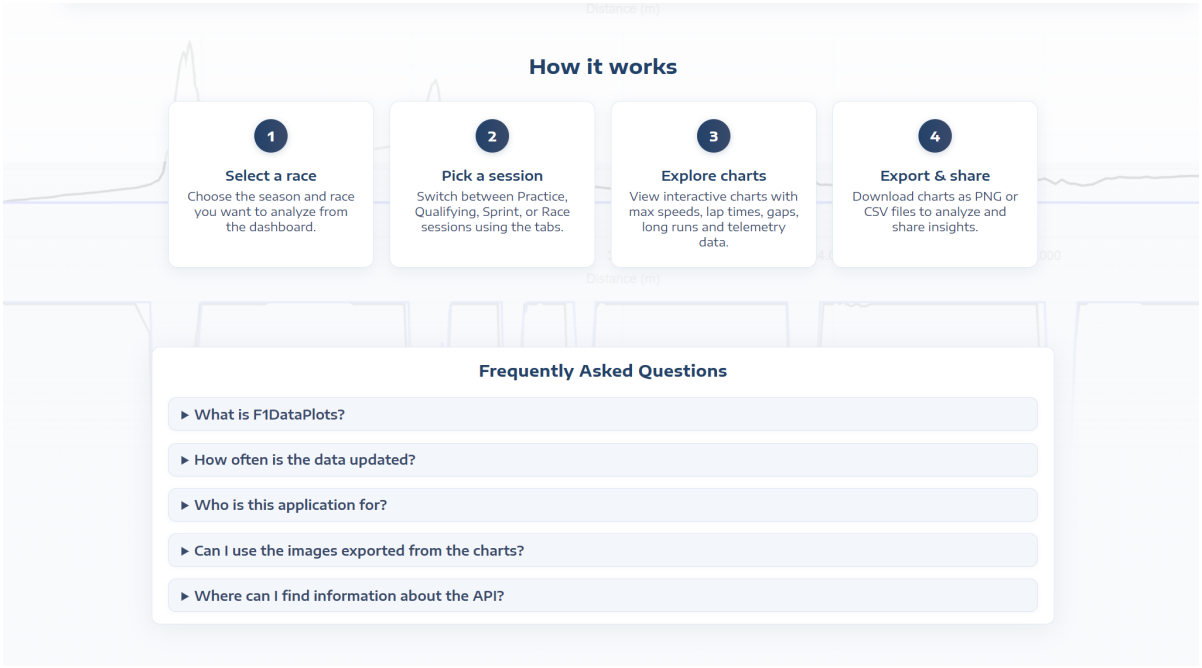
Παρουσίαση του F1dataplots

5.1 Αρχική Σελίδα της Εφαρμογής

Ανοίγοντας την εφαρμογή, ο χρήστης μεταφέρεται στην αρχική σελίδα. Όπως φαίνεται στο Σχήμα 5.1, αρχικά εμφανίζεται ένα μήνυμα καλωσορίσματος. Στο κέντρο της σελίδας υπάρχουν δύο κουμπιά, τα οποία οδηγούν τον χρήστη είτε στο Dashboard είτε στην τεκμηρίωση του API. Στο κάτω μέρος της σελίδας παρουσιάζεται ένας σύντομος οδηγός για το πώς μπορεί ο χρήστης να δημιουργήσει ένα γράφημα, καθώς και μια ενότητα FAQ με απαντήσεις σε συχνές ερωτήσεις. (Σχήμα 5.2).



Σχήμα 5.1: Αρχική Σελίδα του «F1dataplots»



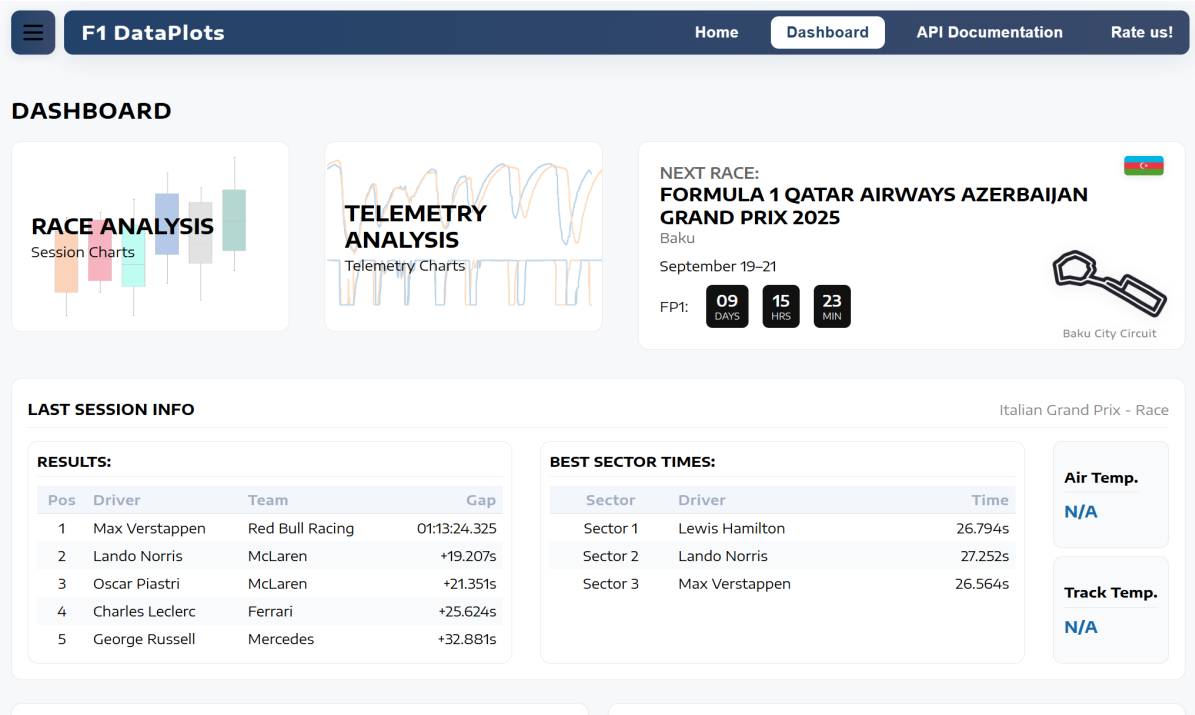
Σχήμα 5.2: Οδηγός χρήσης και FAQ του «F1dataplots»

5.2 Το Dashboard της εφαρμογής

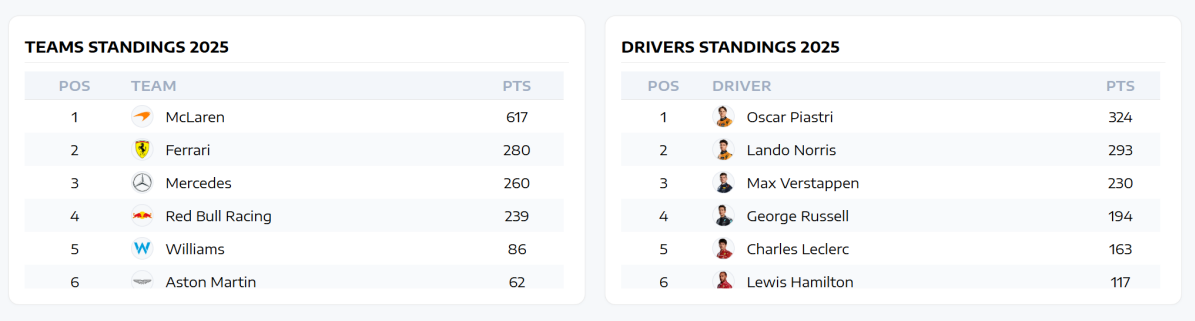
Όταν ο χρήστης επιλέξει το dashboard, μεταφέρεται σε μια σελίδα που συγκεντρώνει τις βασικές δυνατότητες της εφαρμογής, όπως φαίνεται στο Σχήμα 5.3. Στο επάνω μέρος της σελίδας βρίσκεται το μενού πλοήγησης, μέσω του οποίου ο χρήστης μπορεί να μεταβεί στην αρχική σελίδα (Home) ή στην τεκμηρίωση του API (API Documentation), καθώς και να διαχειριστεί τις βασικές του επιλογές.

Η σελίδα χωρίζεται νοητά σε τρεις ενότητες:

- **Πρώτη ενότητα**: Εδώ παρουσιάζονται οι δύο κύριες λειτουργίες της εφαρμογής. Ο χρήστης μπορεί είτε να δει το «Race Analysis», το οποίο παρέχει γραφήματα για κάθε αγώνα, είτε να επιλέξει το «Telemetry Analysis», όπου έχει τη δυνατότητα να δημιουργήσει διαδραστικά γραφήματα σύγκρισης δεδομένων τηλεμετρίας. Δίπλα σε αυτές τις επιλογές εμφανίζεται ένα info-box, το οποίο περιλαμβάνει πληροφορίες για το επόμενο Grand Prix (όπως ημερομηνία διεξαγωγής, χώρα, διάγραμμα της πίστας κ.ά.).
- **Δεύτερη ενότητα**: Ο χρήστης εδώ βλέπει στοιχεία που αφορούν την τελευταία διαδικασία που έχει πραγματοποιηθεί. Συγκεκριμένα, εμφανίζονται τα αποτελέσματα, οι καλύτεροι χρόνοι στα επιμέρους τμήματα της πίστας, καθώς και ορισμένα μετεωρολογικά δεδομένα.
- **Τρίτη ενότητα**: Στο κάτω μέρος της σελίδας παρουσιάζονται οι βαθμολογίες του πρωταθλήματος, τόσο για τους οδηγούς όσο και για τους κατασκευαστές.



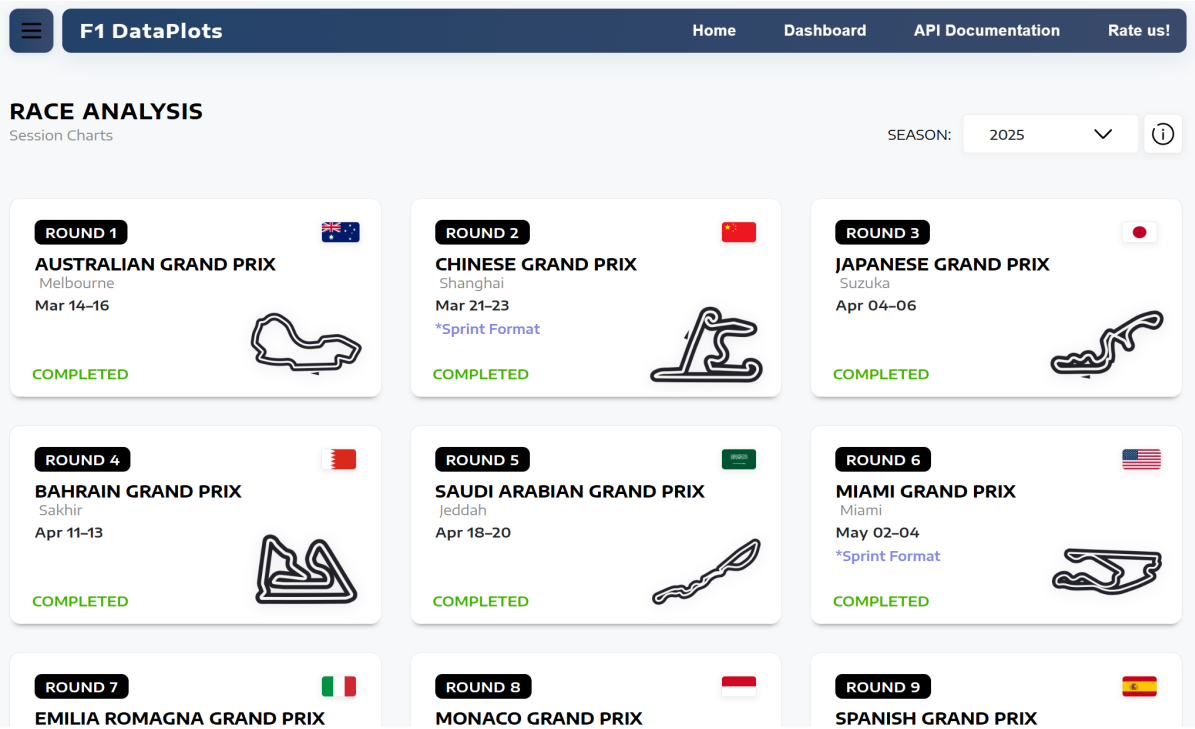
Σχήμα 5.3: Το Dashboard της εφαρμογής (Α΄Μέρος)



Σχήμα 5.4: Το Dashboard της εφαρμογής (Β΄ Μέρος)

5.2.1 Σελίδα Race Analysis της εφαρμογής

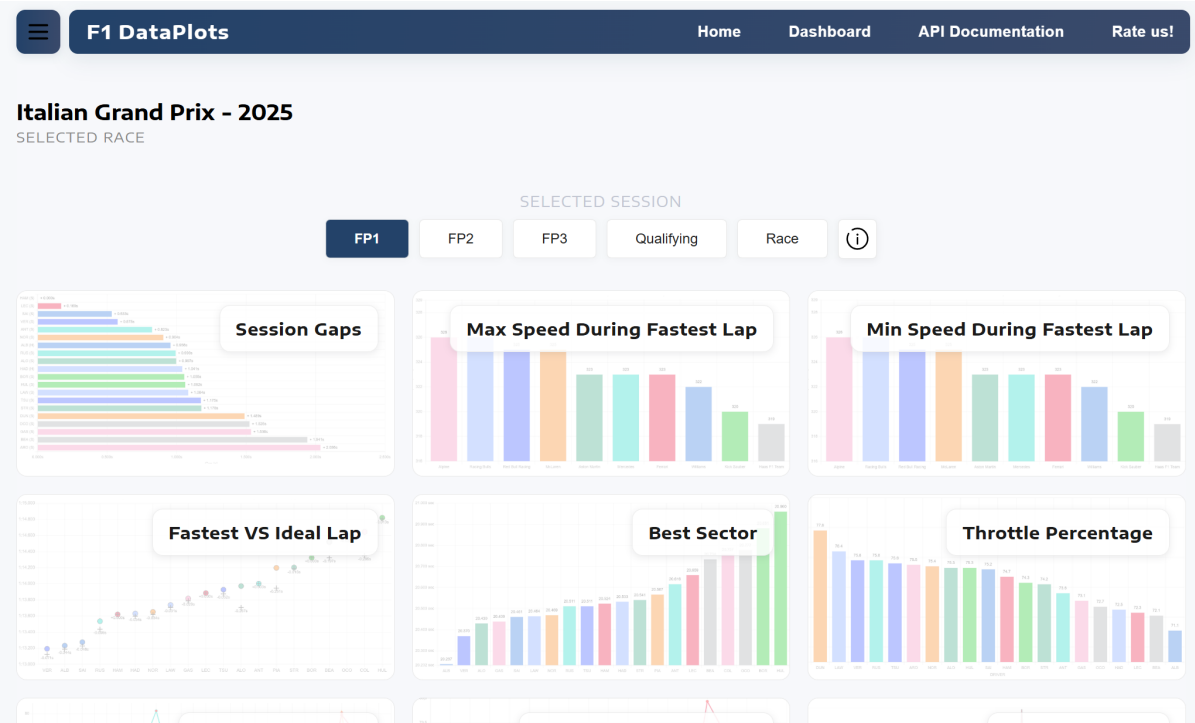
Στην επιλογή «Race Analysis», ο χρήστης μπορεί να δει όλους τους διαθέσιμους αγώνες της επιλεγμένης σεζόν, όπως φαίνεται στο Σχήμα 5.5. Ο χρήστης μπορεί εύκολα να αλλάξει σεζόν μέσω ενός drop-down μενού που βρίσκεται στο επάνω δεξί μέρος της οθόνης. Ακριβώς δίπλα υπάρχει το info-button, το οποίο παρέχει στον χρήστη χρήσιμες πληροφορίες σχετικά με το περιεχόμενο της σελίδας και τις διαθέσιμες λειτουργίες. Κάθε αγώνας παρουσιάζεται με όμορφο και διαδραστικό τρόπο, προσφέροντας στον χρήστη μια ευχάριστη και λειτουργική εμπειρία πλοήγησης.



Σχήμα 5.5: Σελίδα Race Analysis της εφαρμογής

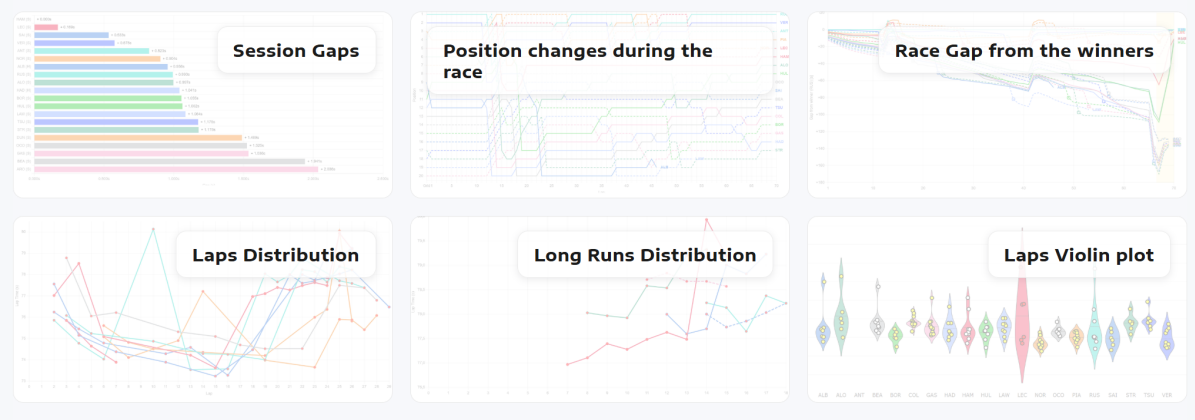
5.2.2 Σελίδα Analysis της εφαρμογής

Αφού ο χρήστης επιλέξει τον αγώνα που επιθυμεί, κάνοντας απλά κλικ πάνω στον αγώνα, μεταφέρεται στην ενότητα «Analysis» του αγώνα (Σχήμα 5.6). Εκεί, αρχικά εμφανίζονται οι διαθέσιμες διαδικασίες του αγώνα με μορφή tabs, ενώ ακριβώς από κάτω παρουσιάζονται τα αντίστοιχα διαθέσιμα γραφήματα. Ανάλογα με τη διαδικασία που έχει επιλεγεί από τα tabs, προβάλλονται διαφορετικά γραφήματα. Για παράδειγμα, στο Σχήμα 5.7 φαίνονται τα γραφήματα που είναι διαθέσιμα για την πρώτη περίοδο ελεύθερων δοκιμών, ενώ στο Σχήμα 5.8 απεικονίζονται εκείνα που αντιστοιχούν στον αγώνα. Επιπλέον, σε περίπτωση που για κάποιο γράφημα δεν υπάρχουν διαθέσιμα δεδομένα, το συγκεκριμένο γράφημα δεν είναι ενεργό προς επιλογή και εμφανίζεται κατάλληλο μήνυμα, όπως φαίνεται στο Σχήμα 5.9.

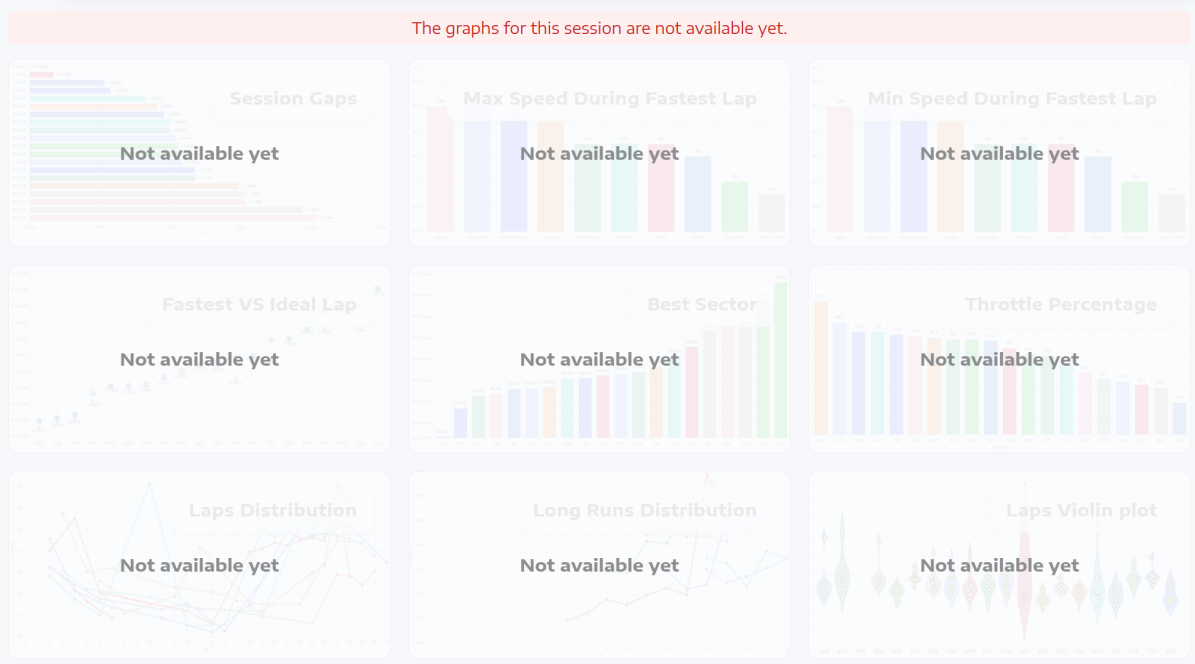


Σχήμα 5.6: Σελίδα Analysis ενός GP της εφαρμογής

Σχήμα 5.7: Γραφήματα του FP1



Σχήμα 5.8: Γραφήματα του αγώνα

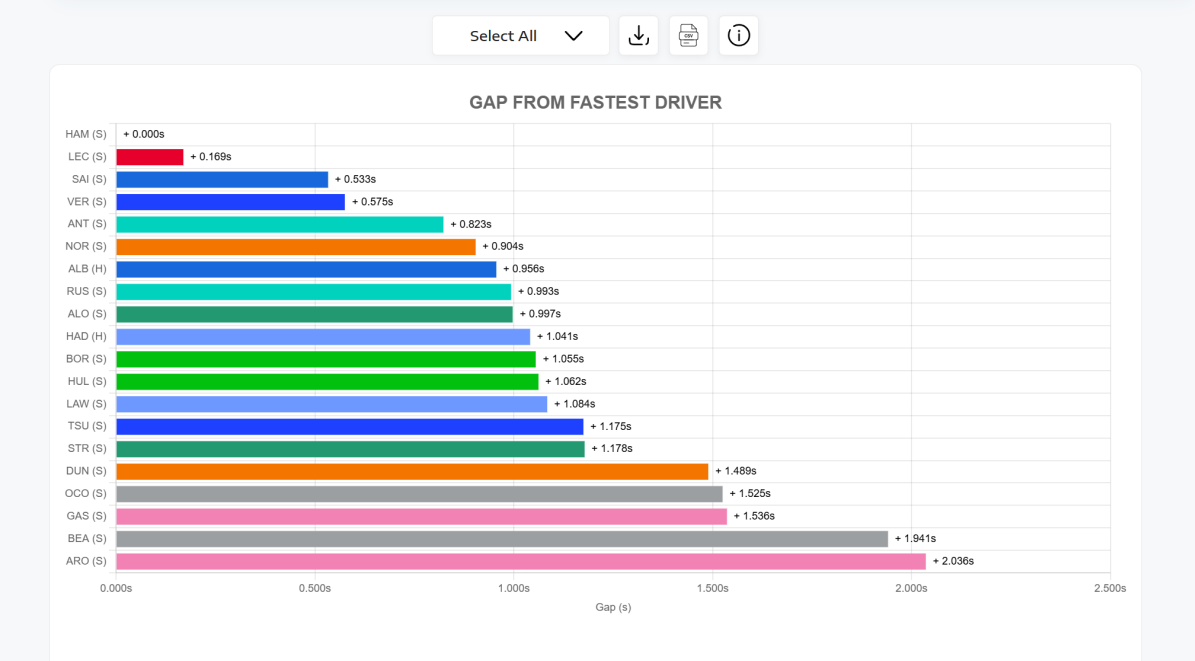


Σχήμα 5.9: Μη διαθέσιμα γραφήματα

5.2.3 Τα γραφήματα της εφαρμογής

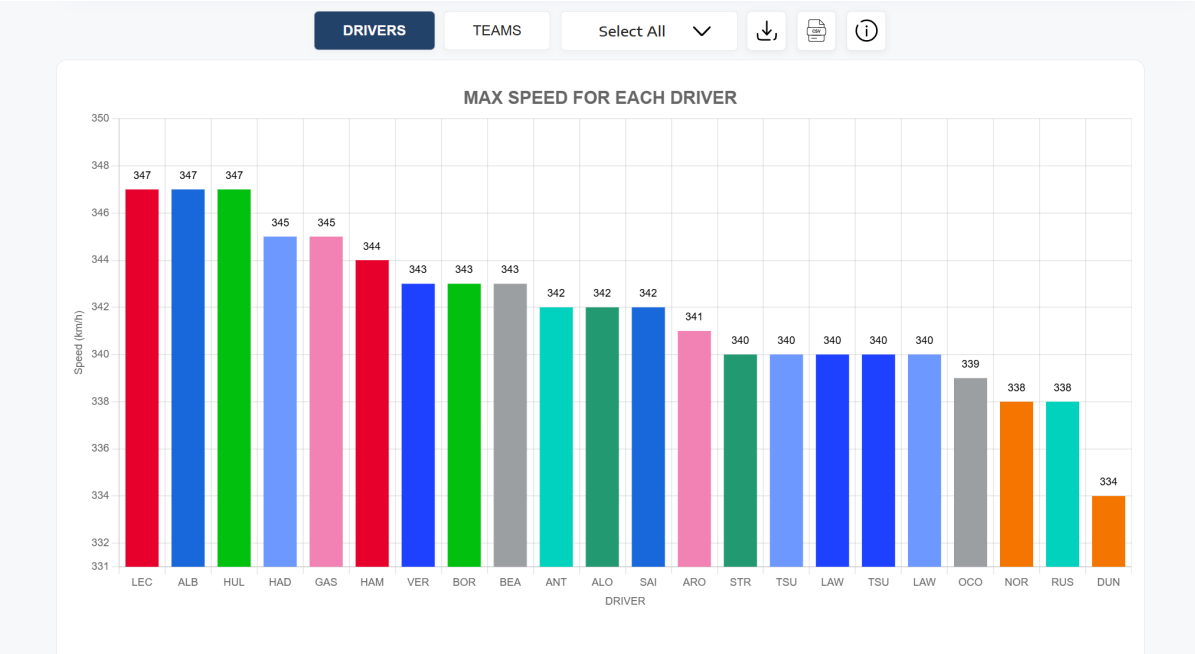
Στο γράφημα «Session Gaps» (Σχήμα 5.10), ο χρήστης μπορεί να δει, για την αντίστοιχη επιλεγμένη διαδικασία, τη διαφορά στους χρόνους γύρων σε σχέση με τον ταχύτερο γύρο της περιόδου, εκφρασμένη σε δευτερόλεπτα. Μέσω του driver selector έχει τη δυνατότητα να προσθέσει ή να αφαιρέσει οδηγούς από το γράφημα. Επιπλέον, το γράφημα είναι διαδραστικό, συνεπώς περνώντας τον κέρσορα πάνω από κάθε μπάρα, εμφανίζονται αναλυτικές πληροφορίες για την ακριβή χρονική διαφορά, καθώς και για το είδος του ελαστικού που χρησιμοποιήθηκε σε εκείνον τον γύρο. Τέλος, ο χρήστης μπορεί να εξάγει το γράφημα σε μορφή εικόνας PNG ή να κατεβάσει τα δεδομένα

σε αρχείο CSV, ώστε να τα αξιοποιήσει για περαιτέρω ανάλυση.



Σχήμα 5.10: Γράφημα «Sessions Gaps»

Στα γραφήματα «Max και Min Speed During Fastest Lap» (Σχήμα 5.11 και Σχήμα 5.12), ο χρήστης μπορεί να δει τη μέγιστη ή την ελάχιστη τελική ταχύτητα που σημείωσε κάθε οδηγός κατά τη διάρκεια του ταχύτερου γύρου του. Υπάρχει η δυνατότητα αλλαγής προβολής από ομάδες σε οδηγούς, ώστε ο χρήστης να επιλέξει ποια δεδομένα θέλει να εξετάσει πιο αναλυτικά. Επιπλέον, παρέχεται η επιλογή εξαγωγής του γραφήματος σε μορφή εικόνας PNG, καθώς και η λήψη των δεδομένων σε αρχείο CSV για περαιτέρω ανάλυση και αξιοποίηση.

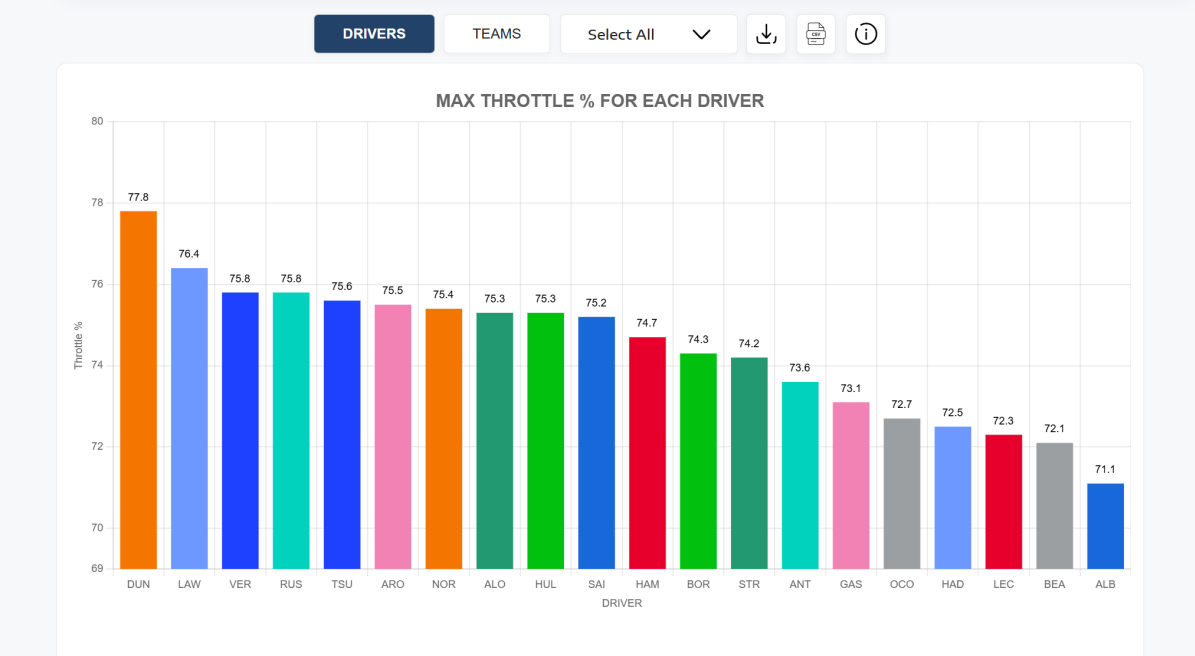


Σχήμα 5.11: Γράφημα «Max Speeds»

Σχήμα 5.12: Γράφημα «Min Speeds»

Στο γράφημα «Max Throttle %», ο χρήστης μπορεί να δει το ποσοστό χρήσης γκαζιού κατά τη διάρκεια της επιλεγμένης διαδικασίας, είτε ανά οδηγό είτε ανά ομάδα, στον ταχύτερο γύρο τους. Μέσω του mode selector μπορεί να εναλλάσσει την προβολή μεταξύ οδηγών και ομάδων, ενώ με το multi-select έχει τη δυνατότητα να επιλέξει ποιοι οδηγοί ή ποιες ομάδες θα εμφανίζονται στο

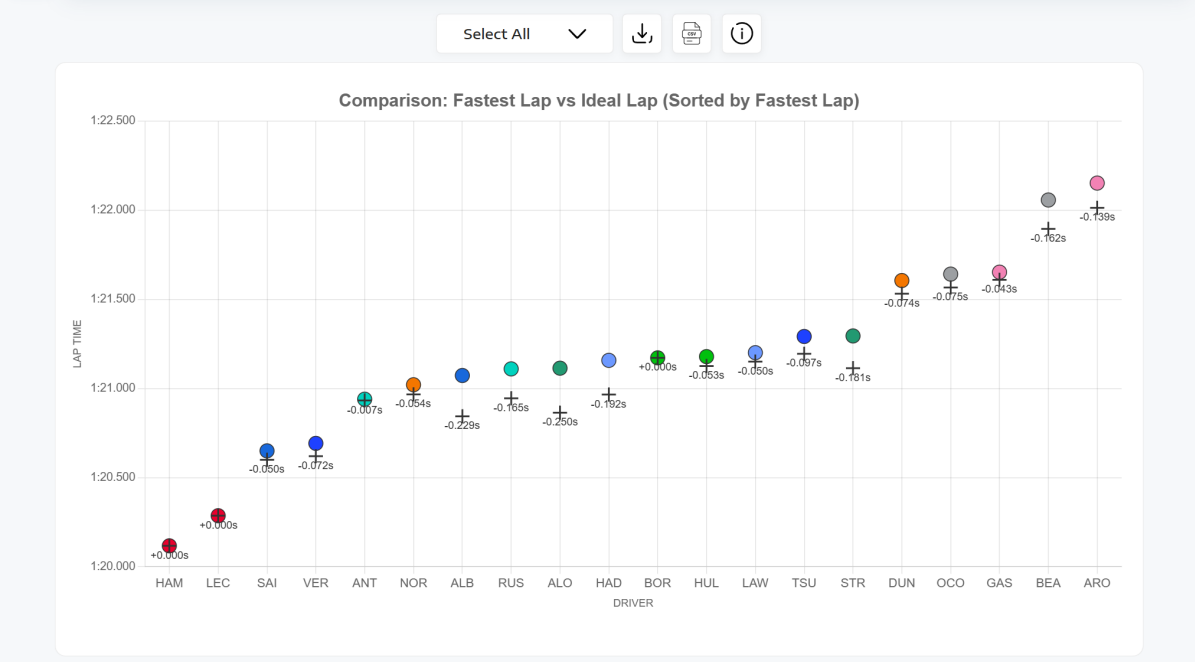
γράφημα. Οι μπάρες είναι χρωματισμένες σύμφωνα με την ομάδα κάθε οδηγού, διευκολύνοντας την οπτική σύγκριση. Επιπλέον, περνώντας τον κέρσορα πάνω από μια μπάρα εμφανίζονται αναλυτικά στοιχεία, όπως ο οδηγός/ομάδα και το ποσοστό γκαζιού. Τέλος, παρέχεται η δυνατότητα εξαγωγής του γραφήματος σε εικόνα PNG ή λήψης των δεδομένων σε αρχείο CSV για περαιτέρω ανάλυση.



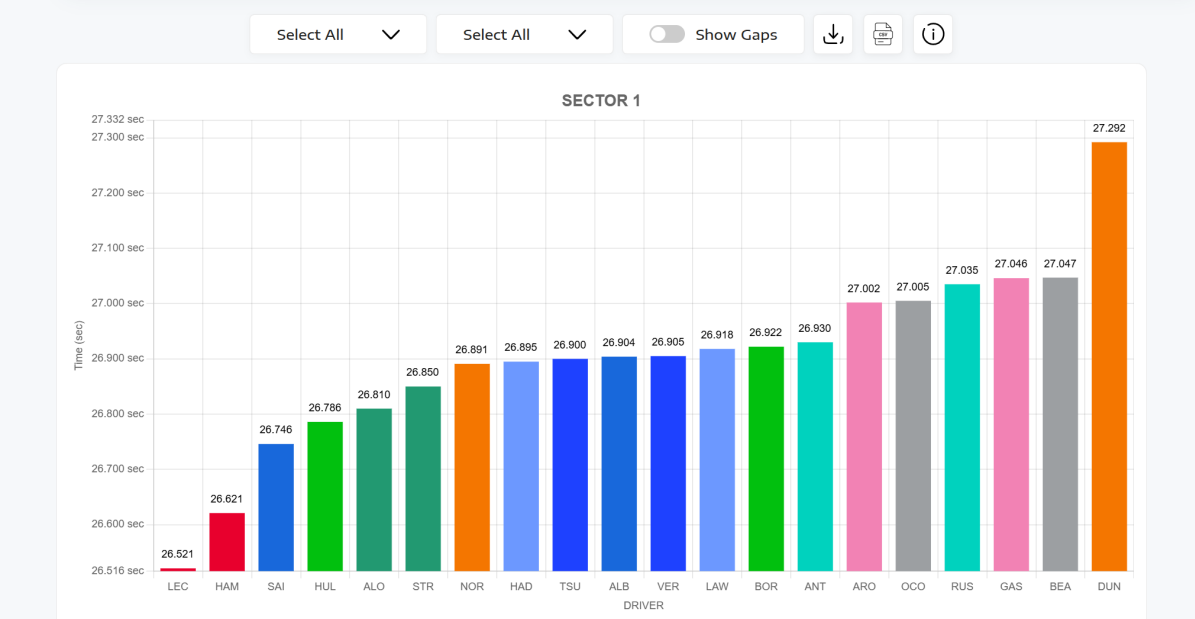
Σχήμα 5.13: Γράφημα «Max Throttle %»

Στο γράφημα «Fastest vs Ideal Lap» (Σχήμα 5.14), ο χρήστης μπορεί να συγκρίνει τον ταχύτερο γύρο που σημείωσαν οι οδηγοί σε μια διαδικασία με τον θεωρητικά ιδανικό γύρο που θα μπορούσαν να πέτυχουν, ο οποίος υπολογίζεται ως το άθροισμα των καλύτερων χρόνων του σε κάθε τομέα της πίστας. Μέσω του driver selector ο χρήστης έχει τη δυνατότητα να προσθέσει ή να αφαιρέσει οδηγούς από το γράφημα. Περνώντας τον κέρσορα πάνω από τα σημεία, εμφανίζονται οι ακριβείς τιμές των χρόνων γύρων μαζί με πρόσθετες λεπτομέρειες. Στην απεικόνιση, ο κύκλος αντιστοιχεί στον πραγματικό ταχύτερο γύρο του οδηγού, ενώ ο σταυρός αναπαριστά τον ιδανικό γύρο που προκύπτει από τους καλύτερους χρόνους σε κάθε τομέα. Τέλος, το γράφημα μπορεί να εξαχθεί σε μορφή εικόνας PNG, ενώ υπάρχει και η δυνατότητα λήψης των δεδομένων σε αρχείο CSV για περαιτέρω ανάλυση.

Στο γράφημα «Best Sector Times» (Σχήμα 5.15), ο χρήστης μπορεί να δει τον ταχύτερο χρόνο που πέτυχε κάθε οδηγός στους επιμέρους τομείς της πίστας. Μέσω του driver selector μπορεί να προσθέσει ή να αφαιρέσει οδηγούς από το γράφημα, ενώ με τον sector selector μπορεί να ρυθμίσει ποιοι τομείς της πίστας (S1, S2, S3) θα εμφανίζονται στο γράφημα. Με την ενεργοποίηση της επιλογής Show Gaps μπορεί να εμφανίσει την διαφορά κάθε οδηγού σε σχέση με τον ταχύτερο σε κάθε τομέα. Καθώς ο χρήστης περνά τον κέρσορα πάνω από τις μπάρες του γραφήματος εμφανίζονται οι ακριβείς χρόνοι και επιπλέον λεπτομέρειες. Υπάρχει η δυνατότητα εξαγωγής του γραφήματος σε εικόνα PNG ή λήψης των δεδομένων σε αρχείο CSV για περαιτέρω ανάλυση.



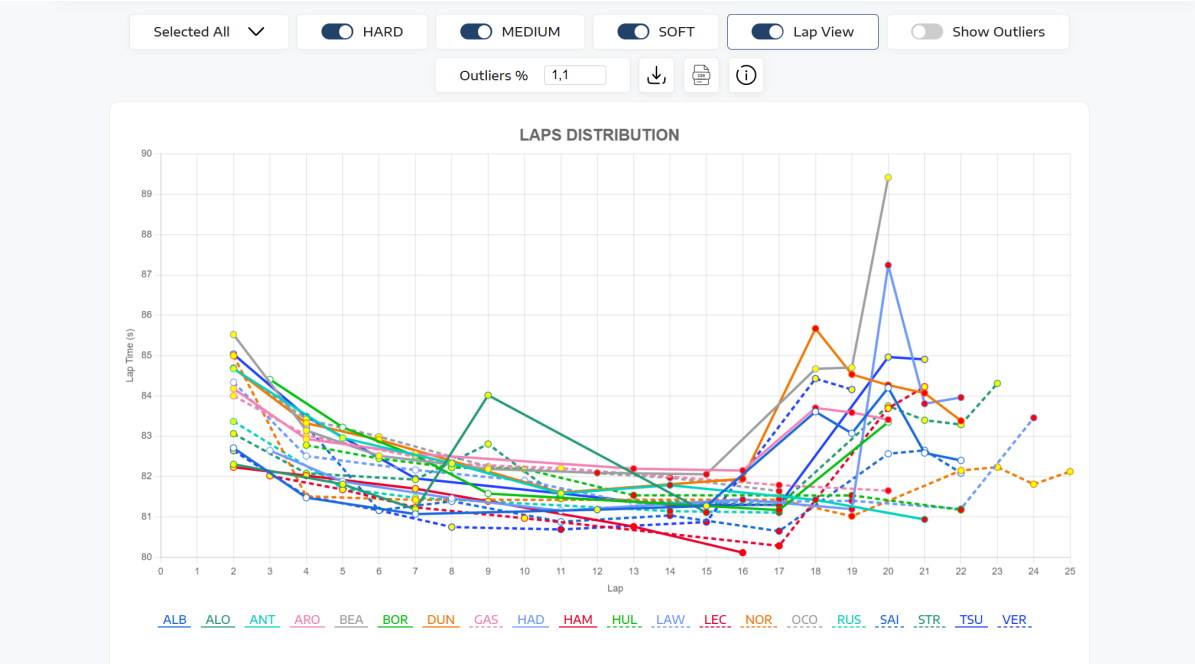
Σχήμα 5.14: Γράφημα «Fastest vs Ideal Lap»



Σχήμα 5.15: Γράφημα «Best Sectors»

Στο γράφημα «Lap Time Distribution» (Σχήμα 5.16), ο χρήστης μπορεί να δει την κατανομή των χρόνων γύρων που έχουν σημειώσει οι οδηγοί σε μια συγκεκριμένη διαδικασία. Υπάρχει η δυνατότητα ενεργοποίησης ή απενεργοποίησης προβολής των ελαστικών (Soft, Medium, Hard κ.λπ.) ώστε να ο χρήστης να μπορεί να αναλύσει τους τύπους ελαστικών που θέλει. Με την επιλογή «Lap View», οι χρόνοι προβάλλονται διαδοχικά ανά γύρο και σε αυτήν τη λειτουργία οι γραμμές που χρησιμοποιούνται είναι συμπαγείς ή διακεκομμένες για να διακρίνονται οι δύο οδηγοί της ίδιας ομάδας (Σχήμα 5.16)). Όταν το «Lap View» είναι απενεργοποιημένο, μπορεί να ενεργοποιηθεί η επιλογή «Show Lines», η οποία συνδέει τα σημεία με γραμμές για καλύτερη οπτική κατανόηση. Παράλληλα, η λειτουργία «Show Outliers» εμφανίζει γύρους που βρίσκονται εκτός του προκαθορισμένου ορίου, το οποίο ρυθμίζεται από την παράμετρο «Outliers %» (π.χ. 110%). Ο χρήστης μπορεί να τροποποιήσει την τιμή αυτή ώστε να αλλάξει το κριτήριο που καθορίζει ποιοι γύροι θεωρούνται ακραίες τιμές. Περνώντας τον κέρσορα πάνω από ένα σημείο, εμφανίζονται αναλυτικές πληροφορίες, όπως ο αριθμός του γύρου, ο χρόνος και το ελαστικό που χρησιμοποιήθηκε. Τέλος, το γράφημα μπορεί να εξαχθεί σε μορφή εικόνας PNG ή τα δεδομένα να κατεβούν σε αρχείο CSV για περαιτέρω ανάλυση.

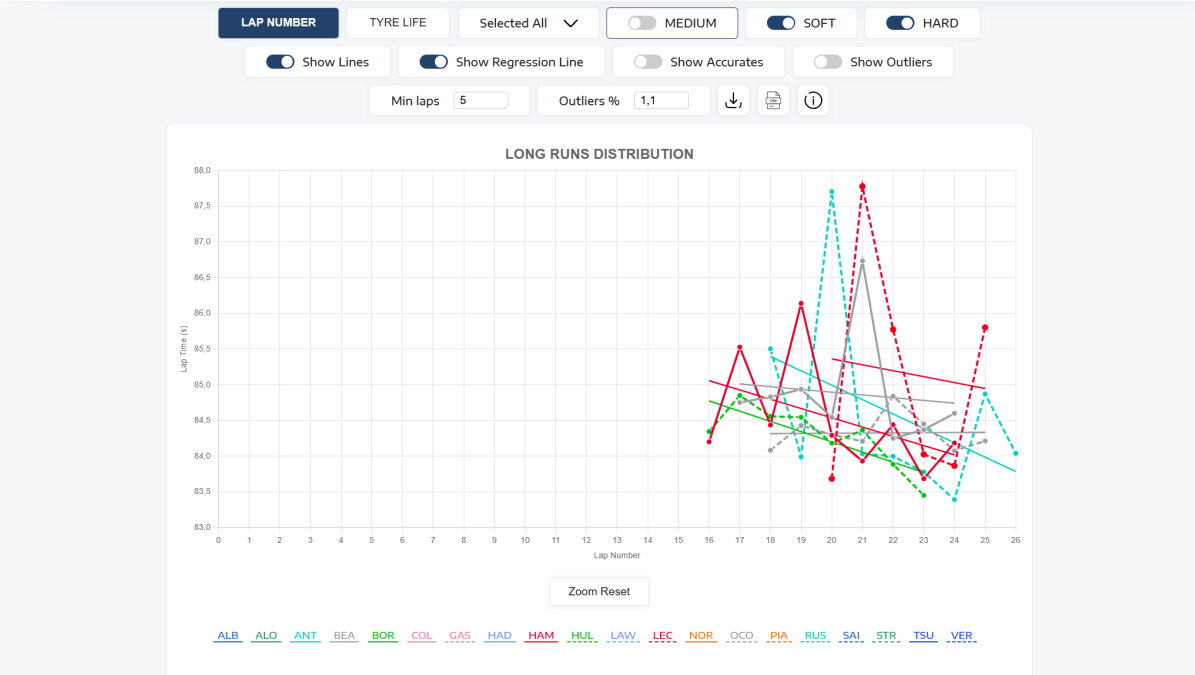
Σχήμα 5.16: Γράφημα «Lap Time Distribution»



Σχήμα 5.17: Γράφημα «Lap Time Distribution» με Lap View

Στο γράφημα «Long Runs Distribution» (Σχήμα 5.18), ο χρήστης μπορεί να δει τις γύρους προσομοιώσεις αγώνα για τους επιλεγμένους οδηγούς. Μέσω του driver selector μπορεί να προσθέσει ή να αφαιρέσει οδηγούς από το γράφημα, ενώ έχει τη δυνατότητα να ενεργοποιήσει ή να απενεργοποιήσει τα διαφορετικά είδη ελαστικών (Soft, Medium, Hard κ.λπ.) ώστε να εστιάσει σε συγκεκριμένα δεδομένα. Ο άξονας X μπορεί να προσαρμοστεί ώστε να εμφανίζει είτε τον αριθμό γύρου είτε τη διάρκεια ζωής των ελαστικών, ανάλογα με την επιλογή του χρήστη. Για καλύτερη κατανόηση των δεδομένων, υπάρχει η επιλογή «Show Lines» για τη σύνδεση των γύρων κάθε προσομοίωσης, καθώς και η επιλογή «Show Regression Line», που προσθέτει γραμμές παλινδρόμησης για κάθε προσομοίωση. Επίσης, με την ενεργοποίηση του «Show Accurates» εμφανίζονται οι έγκυροι γύροι που πληρούν το όριο αλλά δεν εντάσσονται σε κάποιο cluster προσομοίωσης, ενώ με το «Show Outliers» εμφανίζονται οι γύροι εκτός του προκαθορισμένου ορίου, το οποίο ρυθμίζεται μέσω της παραμέτρου «Outliers %» (π.χ. 110%). Η παράμετρος «Min Laps» επιτρέπει τον καθορισμό του ελάχιστου αριθμού συνεχόμενων γύρων που απαιτούνται ώστε μια ακολουθία να θεωρηθεί προσομοίωση αγώνα.

Όπως και στα προηγούμενα γραφήματα έτσι και εδώ ο χρήστης μπορεί να δει αναλυτικά στοιχεία κάθε γύρου (όπως αριθμό γύρου ή διάρκεια ζωής ελαστικού, χρόνο και τύπο ελαστικού) περνώντας τον κέρσορα πάνω από τα δεδομένα. Επιπλέον, κάτω από το γράφημα δημιουργείται ένας πίνακας, όπως φαίνεται στο σχήμα 5.19, που παρουσιάζει όλα τα clusters προσομοίωσης. Στις στήλες του εμφανίζονται το όνομα του οδηγού, ο τύπος του ελαστικού που χρησιμοποιήθηκε, οι αριθμοί των γύρων που περιλαμβάνονται στο cluster, το συνολικό πλήθος γύρων, καθώς και ο μέσος, ο ελάχιστος και ο μέγιστος χρόνος που καταγράφηκε. Τέλος, υπάρχει η δυνατότητα εξαγωγής του γραφήματος σε εικόνα PNG ή των δεδομένων σε αρχείο CSV, για περαιτέρω ανάλυση.



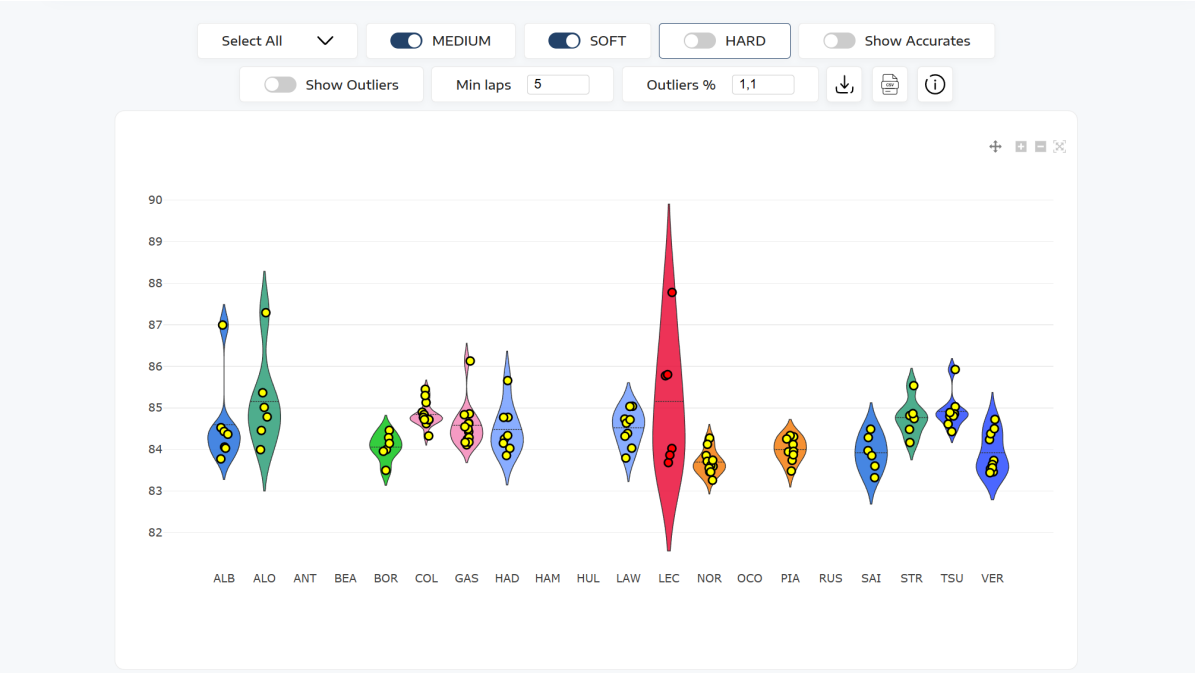
Σχήμα 5.18: Γράφημα «Long Runs Distribution»

Long Runs (Race Sims)

Driver	Compound	Lap Numbers	Total Laps	Mean (s) ▲	Min (s)	Max (s)
HUL	HARD	16, 17, 18, 19, 20, 21, 22, 23	8	84.273	83.451	84.850
OCO	HARD	18, 19, 20, 21, 22, 23, 24, 25	8	84.325	84.075	84.839
HAM	HARD	16, 17, 18, 19, 20, 21, 22, 23, 24	9	84.538	83.685	86.139
RUS	HARD	18, 19, 20, 21, 22, 23, 24, 25, 26	9	84.591	83.392	87.704
BEA	HARD	17, 18, 19, 20, 21, 22, 23, 24	8	84.877	84.250	86.734
LEC	SOFT	20, 21, 22, 23, 24, 25	6	85.155	83.685	87.778

Σχήμα 5.19: Ο πίνακας του γραφήματος «Long Runs Distribution»

Το γράφημα «Long Runs Violin Plot» (Σχήμα 5.20) είναι παρόμοιο με το Long Runs Distribution, με τη διαφορά ότι εδώ τα δεδομένα παρουσιάζονται σε μορφή violin plot, προσφέροντας μια στατιστική απεικόνιση της κατανομής των χρόνων γύρων για κάθε οδηγό και τύπο ελαστικού. Κάθε "violin" απεικονίζει την πυκνότητα και τη διασπορά των χρόνων, επιτρέποντας στον χρήστη να δει πιο καθαρά τη συγκέντρωση και τα άκρα των επιδόσεων. Όπως και στο «Long Runs Distribution», ο χρήστης μπορεί να φιλτράρει οδηγούς ή γόμες, να επιλέξει εμφάνιση ακραίων τιμών (Outliers) και έγκυρων γύρων (Accurates) καθώς και επιλογή του ορίου των ακραίων τιμών.



Σχήμα 5.20: Γράφημα «Long Runs Violin Plot»

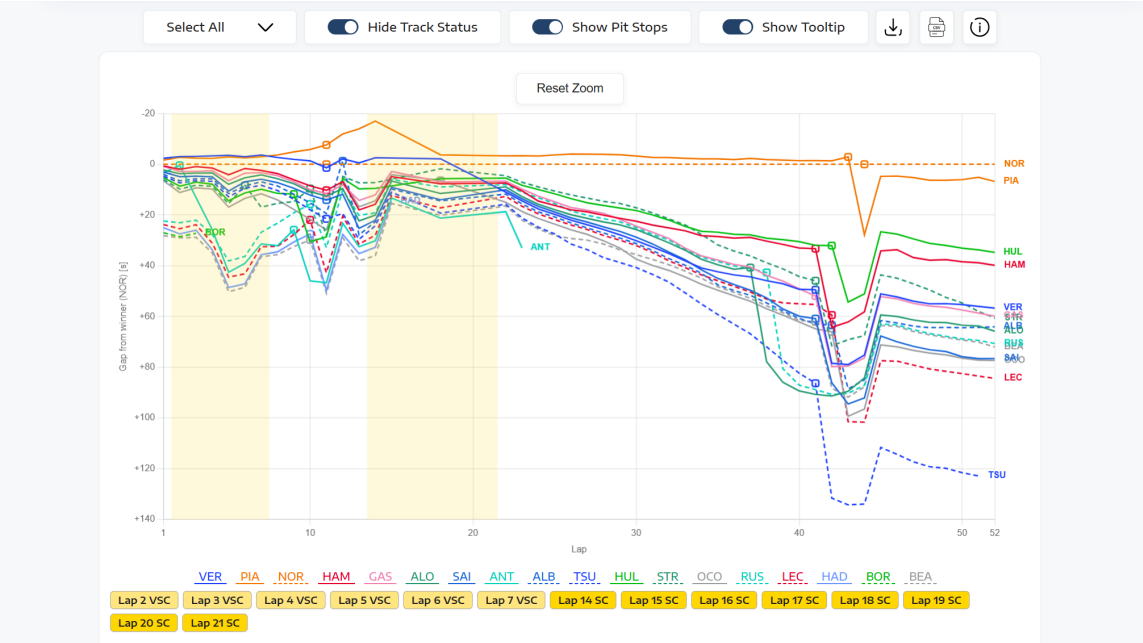
Τα περισσότερα από τα παραπάνω γραφήματα είναι διαθέσιμα τόσο για τις ελεύθερες δοκιμές όσο και για τις κατατακτήριες. Η βασική διαφορά είναι ότι τα γραφήματα που σχετίζονται με τις προσομοιώσεις αγώνα δεν εμφανίζονται στις κατατακτήριες, καθώς δεν πραγματοποιούνται τέτοιες προσομοιώσεις. Αντίθετα, στα ήδη υπάρχοντα γραφήματα προστίθενται δύο επιπλέον για τον αγώνα: το Race Positions Per Lap, το οποίο απεικονίζει τη θέση κάθε οδηγού σε κάθε γύρο, και το Gap From Winner Per Lap, που παρουσιάζει τη διαφορά κάθε οδηγού από τον νικητή γύρο με γύρο.

Στο γράφημα «Race Positions Per Lap» (Σχήμα 5.21), ο χρήστης μπορεί να παρακολουθήσει την εξέλιξη της θέσης κάθε οδηγού γύρο με γύρο κατά τη διάρκεια του αγώνα. Κάθε γραμμή δείχνει την πορεία ενός οδηγού, ενώ οι συμπαγείς και διακεκομμένες γραμμές χρησιμοποιούνται για να διακρίνονται οι δύο οδηγοί της ίδιας ομάδας. Ο κάθετος άξονας δείχνει τη θέση στον αγώνα (1η, 2η, 3η κ.λπ.), ενώ ο οριζόντιος αντιστοιχεί στον αριθμό γύρου. Με το driver selector ο χρήστης μπορεί να προσθέσει ή να αφαιρέσει οδηγούς από το γράφημα. Επιπλέον, περνώντας τον κέρσορα πάνω από ένα σημείο εμφανίζονται λεπτομέρειες, όπως ο οδηγός, ο γύρος και η θέση του εκείνη τη στιγμή. Τέλος, το γράφημα μπορεί να εξαχθεί ως εικόνα PNG, ενώ τα δεδομένα του μπορούν να ληφθούν σε αρχείο CSV για περαιτέρω ανάλυση.

Στο γράφημα «Gap From Winner Per Lap» (Σχήμα 5.22), ο χρήστης μπορεί να δει τη χρονική διαφορά κάθε οδηγού σε σχέση με τον πρωτοπόρο του αγώνα, γύρο με γύρο. Ο νικητής εμφανίζεται πάντα ως επίπεδη γραμμή στα 0 δευτερόλεπτα, ενώ κάθε άλλη γραμμή δείχνει την διαφορά ενός οδηγού σε κάθε γύρο. Μέσω του driver selector ο χρήστης μπορεί να προσθέσει ή να αφαιρέσει οδηγούς. Επιπρόσθετα, υπάρχει η δυνατότητα ενεργοποίησης του «Show Track Status» για την απεικόνιση περιόδων Safety Car (SC) και Virtual Safety Car (VSC), καθώς και του Show Pit Stops, ώστε να σημειώνονται οι γύροι αλλαγής ελαστικών με κυκλικά σύμβολα στη γραμμή κάθε οδηγού.

Σχήμα 5.21: Γράφημα «Race Positions Per Lap»

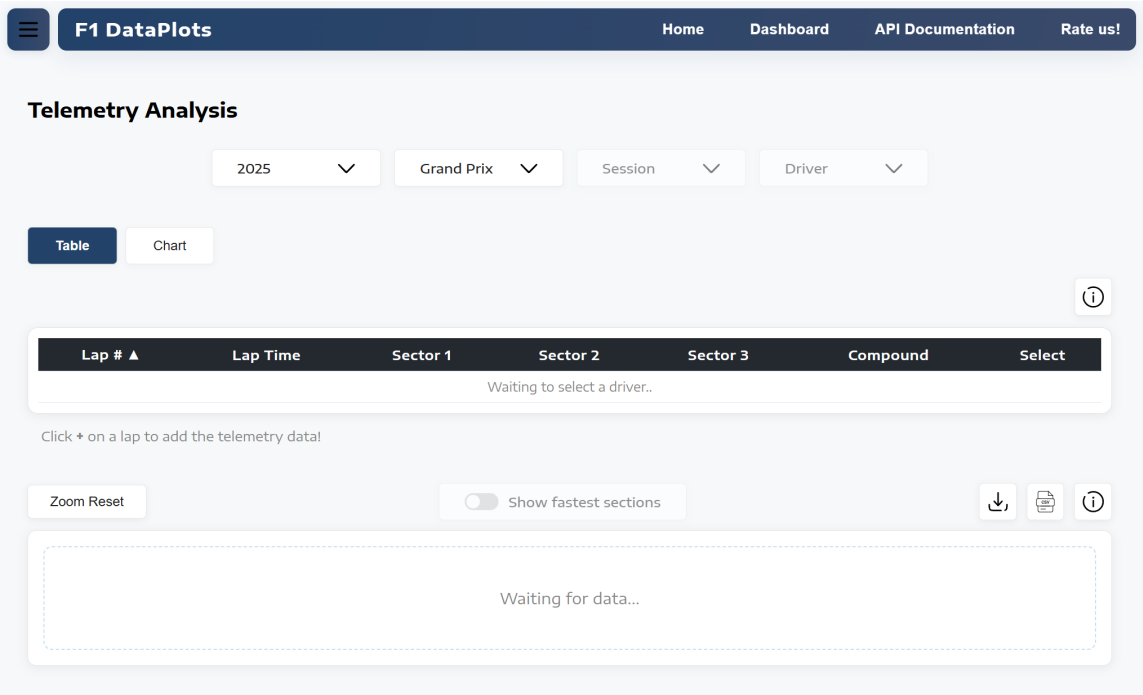
Περνώντας τον κέρσορα πάνω από ένα σημείο στο γράφημα προβάλλονται λεπτομέρειες όπως ο οδηγός, ο γύρος, η διαφορά από τον πρωτοπόρο και τυχόν ενδείξεις πιτ στοπ. Τέλος, το γράφημα μπορεί να εξαχθεί σε εικόνα PNG, ενώ τα δεδομένα του μπορούν να κατεβούν σε αρχείο CSV για περαιτέρω ανάλυση.



Σχήμα 5.22: Γράφημα «Gap From Winner Per Lap»

5.2.4 Σελίδα Telemetry Analysis της εφαρμογής

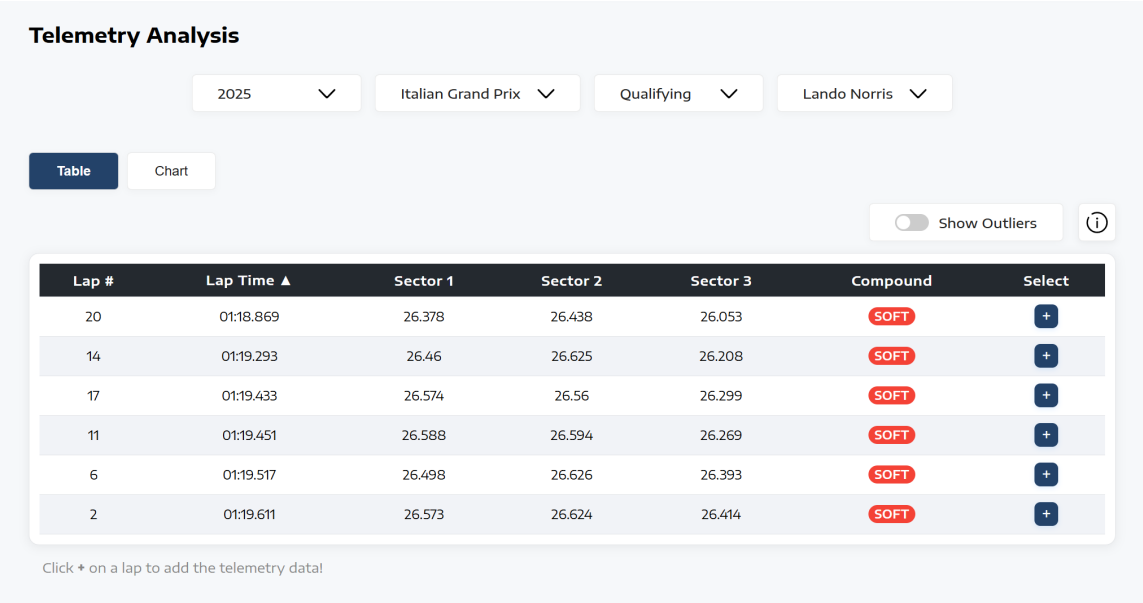
Επιστρέφοντας στο «Dashboard» και επιλέγοντας το «Telemetry Analysis», ο χρήστης μεταφέρεται στην ενότητα που αφορά την ανάλυση τηλεμετρίας (Σχήμα 5.23). Η τηλεμετρία περιλαμβάνει δεδομένα όπως η ταχύτητα, το ποσοστό χρήσης γκαζιού, η πέδηση, η χρήση του DRS, η σχέση στο κιβώτιο ταχυτήτων και οι στροφές ανά δευτερόλεπτο του κινητήρα. Στα σχετικά γραφήματα ο χρήστης μπορεί να συγκρίνει τις τιμές αυτών των δεδομένων για συγκεκριμένους γύρους. Η σύγκριση μπορεί να γίνει είτε μεταξύ διαφορετικών οδηγών, ώστε να αναλυθούν οι διαφορές στον τρόπο οδήγησης, είτε μεταξύ γύρων του ίδιου οδηγού, ώστε να εντοπιστούν διαφορές στην απόδοσή του σε διαφορετικές προσπάθειες. Η ανάλυση πραγματοποιείται σε συνάρτηση με την απόσταση στην πίστα, γεγονός που επιτρέπει την αντιπαραβολή των δεδομένων σε κάθε σημείο της διαδρομής.



Σχήμα 5.23: Σελίδα Telemetry Analysis της εφαρμογής

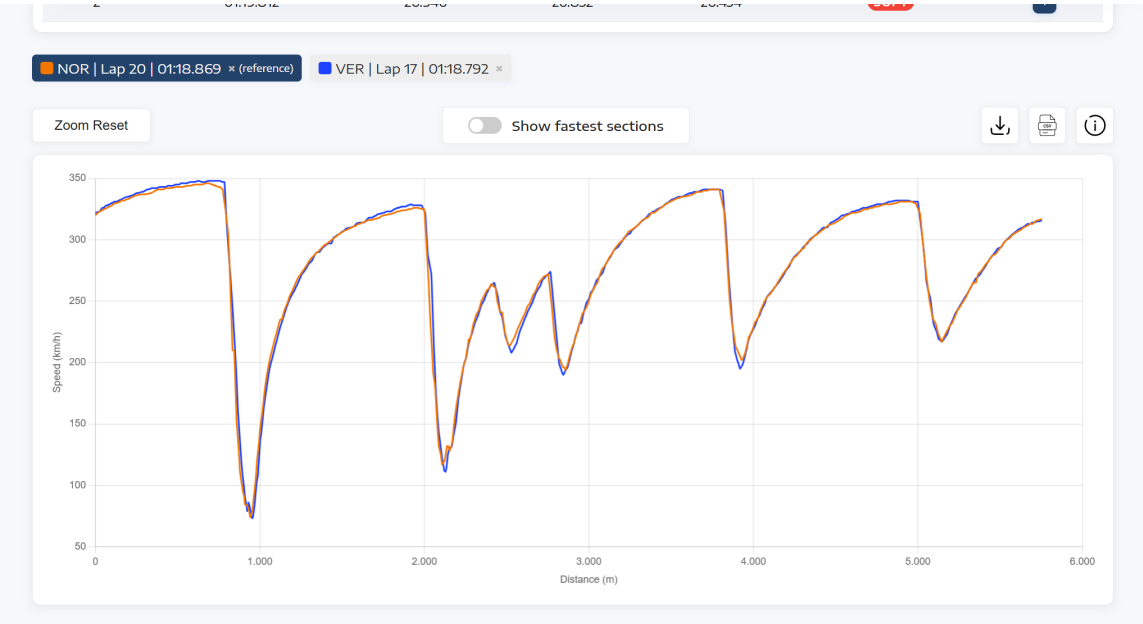
Αρχικά, ο χρήστης καλείται να επιλέξει το έτος, τον αγώνα, τη διαδικασία και τέλος τον οδηγό. Αφού ολοκληρώσει τις επιλογές του, εμφανίζεται ένας πίνακας με όλους τους διαθέσιμους γύρους (Σχήμα 5.24), μαζί με πληροφορίες όπως ο συνολικός χρόνος του γύρου, ο τύπος του ελαστικού και οι χρόνοι στους επιμέρους τομείς της πίστας. Από τον πίνακα αυτόν ο χρήστης μπορεί να επιλέξει έναν ή περισσότερους γύρους, οι οποίοι προστίθενται στο γράφημα που ακολουθεί. Υπάρχει επίσης η δυνατότητα επιλογής γύρων από διαφορετικούς οδηγούς, ώστε η ανάλυση να περιλαμβάνει συγκρίσεις μεταξύ τους.

Οι επιλεγμένοι γύροι εμφανίζονται τόσο στο γράφημα όσο και κάτω από τον πίνακα, όπως φαίνεται στο Σχήμα 5.32, όπου ο χρήστης μπορεί να διαχειριστεί κάθε γύρο ξεχωριστά: να τον αφαιρέσει ή να αλλάξει το χρώμα της γραμμής του για καλύτερη οπτική διάκριση. Ο πρώτος γύρος που επιλέ-



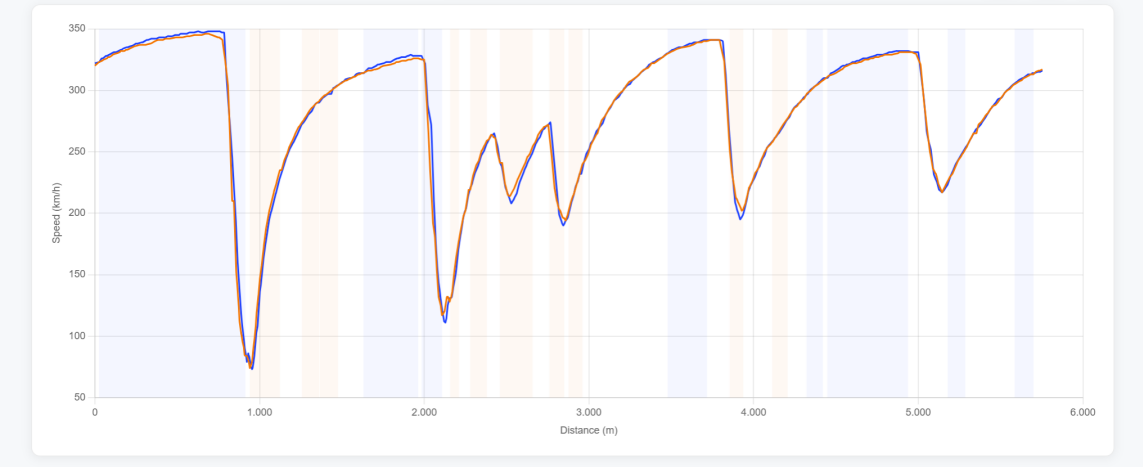
Σχήμα 5.24: Ο πίνακας με τους γύρους του Telemetry Analysis

γεται θεωρείται reference lap, και όλοι οι υπόλοιποι κανονικοποιούνται ως προς αυτόν για σωστή σύγκριση. Επιπλέον, όταν έχουν επιλεγεί ακριβώς δύο γύροι, ο χρήστης μπορεί να ενεργοποιήσει την επιλογή «Show Fastest Sections», ώστε να επισημανθούν τα τμήματα της πίστας όπου κάθε οδηγός είναι ταχύτερος. Τέλος, τα γραφήματα μπορούν να εξαχθούν σε εικόνα PNG, ενώ τα δεδομένα τους μπορούν να κατεβούν σε αρχείο CSV για περαιτέρω ανάλυση.

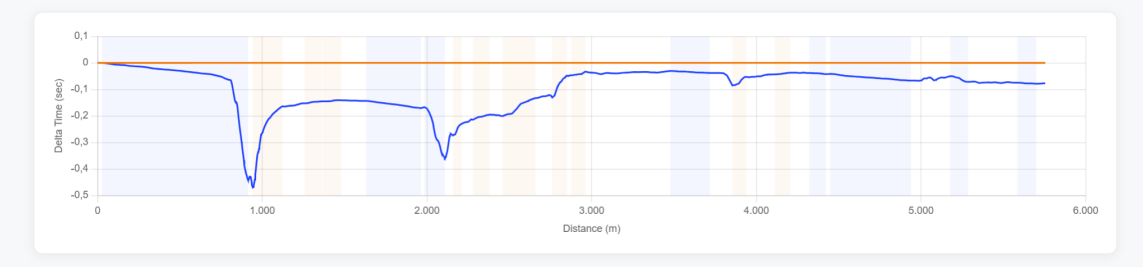


Σχήμα 5.25: Ο πίνακας με τους γύρους του Telemetry Analysis

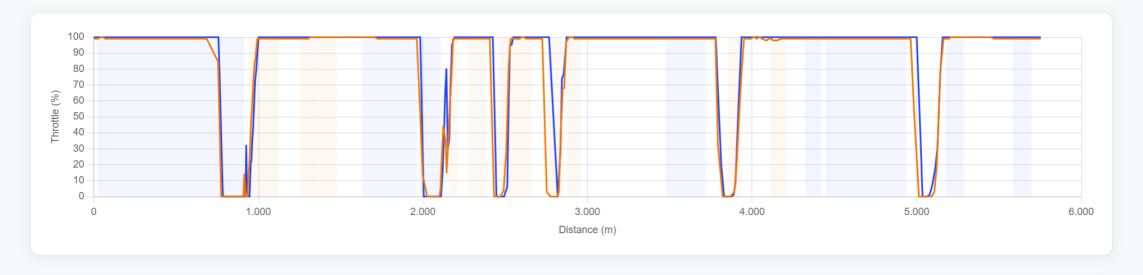
Τα γραφήματα τηλεμετρίας της εφαρμογής είναι συνολικά επτά και εμφανίζονται διαδοχικά, το ένα κάτω από το άλλο, στην ίδια σελίδα. Η διάταξη αυτή έχει σχεδιαστεί έτσι ώστε ο χρήστης να μπορεί να συγκρίνει εύκολα τα δεδομένα και να έχει μια ολοκληρωμένη εικόνα της ανάλυσης. Στα παρακάτω σχήματα παρουσιάζονται όλα τα γραφήματα που είναι διαθέσιμα στον χρήστη.



Σχήμα 5.26: Γράφημα Ταχύτητας

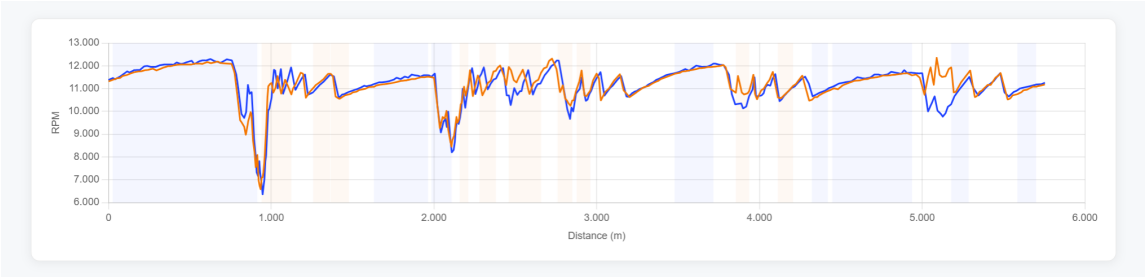


Σχήμα 5.27: Γράφημα διαφοράς χρόνου (Delta Time)

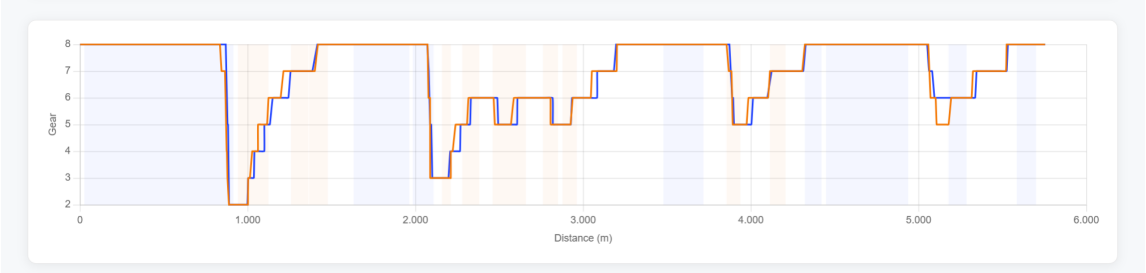


Σχήμα 5.28: Γράφημα Χρήσης Γκαζιού

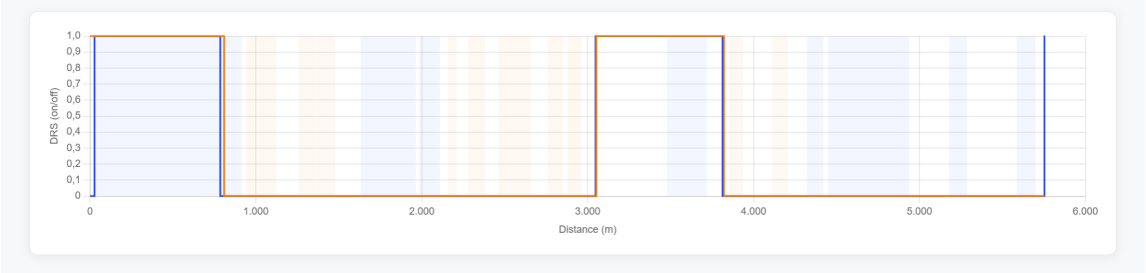
Σχήμα 5.29: Γράφημα Πέδησης



Σχήμα 5.30: Γράφημα στροφών του κινητήρα (Engine RPM



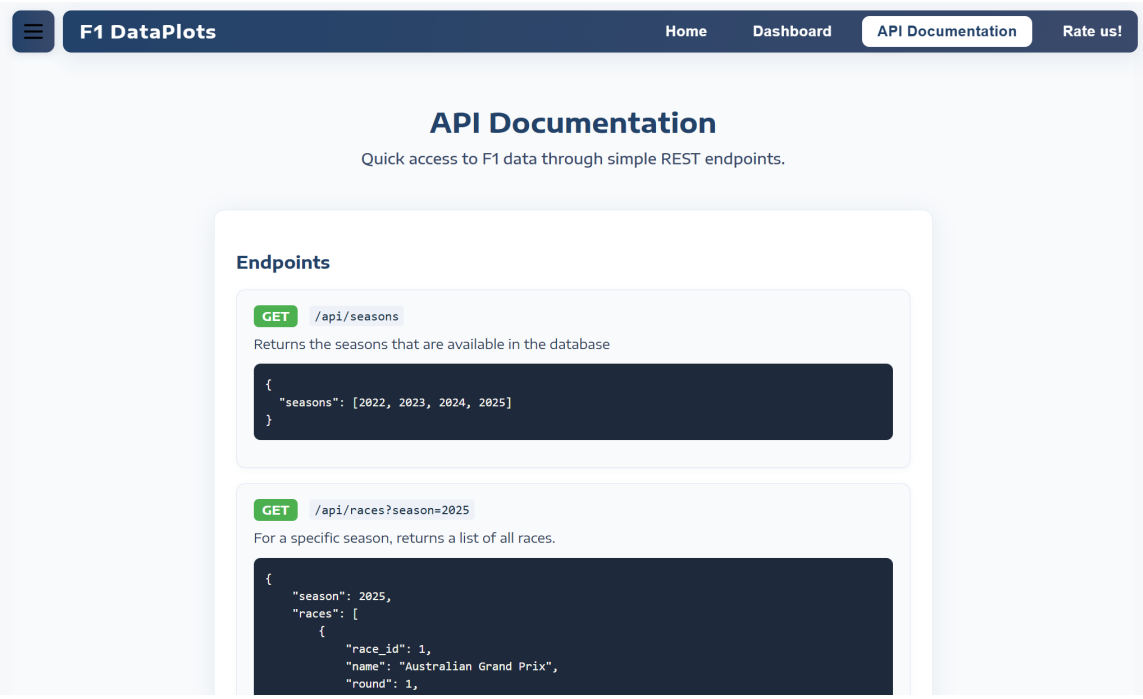
Σχήμα 5.31: Γράφημα σχέσης στο κιβώτιο ταχυτήτων



Σχήμα 5.32: Γράφημα χρήσης του DRS

5.3 Σελίδα API Documentation της εφαρμογής

Τέλος, στη σελίδα API Documentation (Σχήμα 5.33) παρουσιάζονται όλα τα διαθέσιμα endpoints της εφαρμογής, μαζί με περιγραφές για τη λειτουργία τους και παραδείγματα κλήσεων. Με αυτόν τον τρόπο ο χρήστης ή ο προγραμματιστής μπορεί να κατανοήσει εύκολα πώς να αξιοποιήσει το API και να δει τις απαιτούμενες παραμέτρους που χρειάζονται τα endpoints.



Σχήμα 5.33: Σελίδα του API Documentation

Κεφάλαιο 6

Αξιολόγηση

6.1 Το εργαλείο αξιολόγησης SUS (System Usability Scale)

Για την αξιολόγηση της ευχρηστίας της εφαρμογής επιλέχθηκε το System Usability Scale (SUS). Το SUS είναι ένα διεθνές εργαλείο που επιτρέπει τη αξιολόγηση της φιλικότητας και της χρηστικότητας μιας εφαρμογής. Αναπτύχθηκε από τον John Brooke το 1986 στη Digital Equipment Corporation στο Ηνωμένο Βασίλειο. Το SUS αποτελείται από δέκα ερωτήσεις που σχετίζονται με την εμπειρία χρήσης της εφαρμογής, οι οποίες απαντώνται σε μια κλίμακα από 1 (Διαφωνώ Απόλυτα) έως 5 (Συμφωνώ Απόλυτα). Η πλήρης λίστα των ερωτήσεων παρουσιάζεται παρακάτω:

1. I think that I would like to use this website frequently.
2. I found the website unnecessarily complex.
3. I thought the website was easy to use.
4. I think that I would need the support of a technical person to be able to use this website.
5. I found the various functions in this website were well integrated.
6. I thought there was too much inconsistency in this website.
7. I would imagine that most people would learn to use this website very quickly.
8. I found the website very cumbersome to use.
9. I felt very confident using the website.
10. I needed to learn a lot of things before I could get going with this website.

Για τον υπολογισμό του τελικού σκορ SUS, υπάρχει μια συγκεκριμένη διαδικασία. Αρχικά, κάθε απάντηση μετατρέπεται σε μια νέα τιμή. Για τις ερωτήσεις με μονό αριθμό αφαιρείται 1 από την απάντηση, ενώ για τις ερωτήσεις με ζυγό αριθμό η απάντηση αφαιρείται από το 5. Στη συνέχεια, οι νέες τιμές που προκύπτουν αθροίζονται και το αποτέλεσμα πολλαπλασιάζεται με το 2,5, ώστε να προκύψει το συνολικό σκορ σε κλίμακα από 0 έως 100. Εάν η τιμή του τελικού σκορ είναι του 80 ή μεγαλύτερη η εφαρμογή θεωρείται εξαιρετική (A), εάν το σκορ είναι κοντά στο 68 η εφαρμογή χαρακτηρίζεται μέτρια (C), ενώ σκορ κάτω από 51 δείχνει σοβαρά προβλήματα ευχρηστίας (F).

6.2 Αποτελέσματα αξιολόγησης της εφαρμογής

Για την αξιολόγηση της εφαρμογής επιλέχθηκε ένα δείγμα χρηστών με σχετικές γνώσεις γύρω από τον μηχανοκίνητο αθλητισμό και ειδικότερα τη Formula 1, ώστε τα αποτελέσματα να είναι όσο το δυνατόν πιο αξιόπιστα. Συνολικά, απάντησαν 23 συμμετέχοντες και η μέση βαθμολογία που προέκυψε για την εφαρμογή ήταν 85,4. Οι αναλυτικές βαθμολογίες κάθε συμμετέχοντα παρουσιάζονται στον πίνακα 6.1.

Πίνακας 6.1: Πίνακας με τα αποτελέσματα του SUS

User	**Score**
1	80
2	70
3	80
4	100
5	95
6	90
7	95
8	85
9	80
10	80
11	77,5
12	97,5
13	90
14	90
15	95
16	80
17	90
18	92,5
19	65
20	87,5
21	87,5
22	90
23	67,5
Total	**85,4**

Κεφάλαιο 7

Συμπεράσματα και Μελλοντικές Επεκτάσεις

7.1 Συμπεράσματα

Στα πλαίσια της παρούσας διπλωματικής εργασίας, αναπτύχθηκε η διαδικτυακή εφαρμογή «F1dataplots», η οποία αποτελεί ένα εργαλείο οπτικοποίησης και ανάλυσης δεδομένων αγώνων Formula 1. Η εφαρμογή αξιοποιεί δεδομένα από την βιβλιοθήκη FastF1 της Python, τα οποία οπτικοποιούνται μέσα από διαδραστικά γραφήματα και πίνακες, προσφέροντας στους χρήστες τη δυνατότητα να δημιουργήσουν την δικές τους αναλύσεις για την απόδοση των οδηγών και ομάδων. Οι λειτουργίες της καλύπτουν ένα ευρύ φάσμα ανάλυσης, από τη σύγκριση δεδομένων τηλεμετρίας έως την ανάλυση προσομοιώσεων αγώνα και χρήσης των ελαστικών.

Η διεπαφή χρήστη (UI) της εφαρμογής έχει σχεδιαστεί με γνώμονα την απλότητα και την ευχρηστία, ώστε να είναι προσιτή τόσο σε έμπειρους αναλυτές δεδομένων όσο και σε αρχάριους χρήστες με βασικές γνώσεις γύρο από Formula 1. Μέσα από ένα καλά οργανωμένο περιβάλλον, οι χρήστες μπορούν να περιηγηθούν στις κύριες ενότητες, όπως το Dashboard, το Race Analysis, το Telemetry Analysis και το API Documentation, έχοντας πάντα στη διάθεσή τους σαφή καθοδήγηση και εύχρηστα διαδραστικά εργαλεία.

Συμπερασματικά, η «F1dataplots» δίνει τη δυνατότητα σε όλους τους χρήστες να εμβαθύνουν στα δεδομένα της Formula 1 μέσα από απλά και κατανοητά εργαλεία. Η ευχρηστία της εφαρμογής, σε συνδυασμό με τα διαδραστικά γραφήματα και την οργανωμένη παρουσίαση των πληροφοριών, την καθιστούν προσιτή σε όλους.

7.2 Μελλοντικές Επεκτάσεις

Η εφαρμογή «F1dataplots» παρέχει ένα ολοκληρωμένο περιβάλλον οπτικοποίησης δεδομένων Formula 1. Παρ' όλα αυτά η εφαρμογή έχει δυνατότητες βελτίωσης και εξέλιξης. Μερικές από αυτές θα μπορούσαν να είναι:

Ανάπτυξη Εφαρμογής για Κινητές Συσκευές

Δημιουργία αντίστοιχης εφαρμογής για κινητές συσκευές (Android και iOS). Μέσα από ένα mobile app, οι χρήστες θα μπορούσαν να έχουν άμεση πρόσβαση στα γραφήματα και τα δεδομένα τηλεμετρίας από οποιαδήποτε κινητή συσκευή. Ένα τέτοιο περιβάλλον θα προσέφερε μεγαλύτερη ευελιξία και αμεσότητα.

Ενσωμάτωση μηχανισμών μηχανικής μάθησης

Μία ακόμη προσθήκη θα μπορούσε να είναι η ενσωμάτωση μηχανισμών μηχανικής μάθησης για την πρόβλεψη επιδόσεων και στρατηγικών. Μέσω αλγορίθμων ανάλυσης δεδομένων, η εφαρμογή θα μπορούσε να παρέχει εκτιμήσεις για τον ρυθμό φθοράς ελαστικών, τις πιθανές στρατηγικές pit-stops ή ακόμη και για την τελική κατάταξη των οδηγών.

Βιβλιογραφία

[1] F. 1, ``Hall of fame - the world champions." https://www.formula1.com/en/drivers/hall-of-fame. Accessed: 29-07-2025.

[2] F. 1, ``The family tree of f1's 10 teams and how they came to be." https://www.formula1.com/en/latest/article/the-family-tree-of-f1s-10-teams-and-how-they-came-to-be.4CHsJYEuIjG12YVizwSUcJ. Accessed: 29-07-2025.

[3] F. 1, ``Everything you need to know about f1 – drivers, teams, cars, circuits and more." https://www.formula1.com/en/latest/article/drivers-teams-cars-circuits-and-more-everything-you-need-to-know-about.7iQfL3Rivf1comzdqV5jwc. Accessed: 01-08-2025.

[4] M.-A. P. F. O. Team, ``Data and electronics in f1, explained!." https://www.mercedesamgf1.com/news/feature-data-and-electronics-in-f1-explained. Accessed: 01-08-2025.

[5] R. N. Bhambwani, ``Data: The unseen driver in formula 1 cars." https://medium.com/formula-one-forever/data-the-unseen-driver-in-formula-1-cars-63f31c16f2fe. Accessed: 03-08-2025.

[6] M. Applied, ``Advanced telemetry linked acquisition system." https://www.motionapplied.com/products/ATLAS. Accessed: 25-08-2025.

[7] M.-A. P. F. O. Team, ``How does f1 simulation work?." https://www.mercedesamgf1.com/news/how-does-f1-simulation-work. Accessed: 02-08-2025.

[8] S. Codling, *Speed Read F1: The Technology, Rules, History and Concepts Key to the Sport*. Motorbooks / Quarto Publishing Group USA, 2017.

[9] FastF1, ``Fastf1." https://docs.fastf1.dev/index.htmle. Accessed: 30-06-2025.

[10] ``What is python?." https://www.python.org/doc/essays/blurb/. Accessed: 25-07-2025.

[11] M. Wes, *Python for data analysis*. O'Reilly Media, Inc., 1 ed., 2012.

[12] S. Worsley, ``What is python? everything you need to know to get started." https://www.datacamp.com/blog/all-about-python-the-most-versatile-programming-language?dc_referrer=https%3A%2F%2Fwww.google.com%2F. Accessed: 25-07-2025.

[13] S. Worsley, ``Mysql: Understanding what it is and how it's used." https://www.oracle.com/mysql/what-is-mysql/. Accessed: 26-07-2025.

[14] P. L. Technologies, *Beginning PHP & MySQL Development: Code Your Own Dynamic Website Today*. North Charleston, SC, USA: CreateSpace Independent Publishing Platform, 2014.

[15] ``Mysql: Understanding what it is and how it's used." https://dev.mysql.com/doc/refman/8.4/en/what-is-mysql.html. Accessed: 26-07-2025.

[16] ``What is php and what can it do?." https://www.php.net/manual/en/introduction.php. Accessed: 27-07-2025.

[17] Ingario, ``Server-side scripting with php: An overview." https://clouddevs.com/php/server-side-scripting/. Accessed: 27-07-2025.

[18] D. F. Harshil Agrawal, ``What is typescript and why should you use it?." https://www.contentful.com/blog/what-is-typescript-and-why-should-you-use-it/. Accessed: 19-08-2025.

[19] J. Olawanle, ``Typescript vs javascript: How are they different?." https://hygraph.com/blog/typescript-vs-javascript. Accessed: 19-08-2025.

[20] R. H. Jansen, *Learning TypeScript*. Packt Publishing, 2015.

[21] ``React." https://react.dev/. Accessed: 20-08-2025.

[22] C. Gackenheimer, *Introduction to React*. USA: Apress, 1st ed., 2015.

[23] ``Jsx." https://www.typescriptlang.org/docs/handbook/jsx.html. Accessed: 21-08-2025.

[24] ``Vite - overview." https://vite.dev/guide/. Accessed: 21-08-2025.

[25] E. Deniz, ``The build tool for modern web development." https://medium.com/@emre.deniz/vite-the-build-tool-for-modern-web-development-0e99906f2f0c. Accessed: 21-08-2025.

[26] A. Becker, ``What is heidisql?." https://www.heidisql.com/. Accessed: 10-08-2025.