



Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης
Πολυτεχνική Σχολή
Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Τηλεπικοινωνιών

Τίτλος διπλωματικής

Διπλωματική Εργασία
του
Θεοφάνη Θαρρόπουλου

Επιβλέπων: Ανδρέας Συμεωνίδης
Καθηγητής Α.Π.Θ.

6 Οκτωβρίου 2024

Περίληψη

Αντικείμενο της παρούσας διπλωματικής εργασίας αποτελεί η έρευνα για την αξιολόγηση της ποιότητας του κώδικα που παράγεται από Μεγάλα Γλωσσικά Μοντέλα (LLMs), και πιο συγκεκριμένα από το GitHub Copilot[2]. Η μελέτη εστιάζει στην αξιολόγηση της ποιότητας του κώδικα που παράγεται από το όπιλοτ και στην βελτιστοποίηση των προτροπών (prompts) για την επίτευξη των επιθυμητών αποτελεσμάτων μέσω τεχνικών μηχανικής προτροπής (prompt engineering) και της μηχανικής μάθησης. Τα αποτελέσματα αποδεικνύουν τις δυνατότητες και τους περιορισμούς του όπιλοτ στην παραγωγή ποιοτικού κώδικα και προσφέρουν νέες προσεγγίσεις για την βελτίωση της αλληλεπίδρασης μεταξύ του χρήστη και του εργαλείου μέσω στοχευμένων τεχνικών προτροπής.

Abstract

Empty

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέποντα της διπλωματικής εργασίας μου, κ. Ανδρέα Συμεωνίδη, για την ευκαιρία που μου έδωσε να εργαστώ πάνω σε ένα θέμα που με ενθουσιάζει και για την υποστήριξη που μου προσέφερε καθ' όλη την διάρκεια της εργασίας μου, καθώς και για τη βοήθεια σε όσες δυσκολίες βίωσα κατά την έρευνα. Επίσης, θα ήθελα να ευχαριστήσω τους αγαπημένους μου φίλους, Θοδωρή, Παρασκευά, Λεωνίδα, Βασίλη, Ιορδάνη, Λευτέρη, Θεοδόση, για την υπομονή και στήριξη που έδιξαν όλο αυτό τον καιρό. Θα ήθελα να ευχαριστήσω την αγαπημένη μου μητέρα, Βασιλική, για τις θυσίες της όλα αυτά τα χρόνια, την αγάπη και δύναμη που μου έχει δώσει από τη μέρα που γεννήθηκα, τον αγαπημένο μου πατέρα Χρήστο, που με γέμιζε πάντοτε χαρά με την περιφάνεια του για μένα.

Τέλος, θα ήθελα να ευχαριστήσω περισσότερο από όλους, τον μοναδικό μου αδερφό Γιώργο, το άτομο που αγαπώ και θαυμάζω περισσότερο από καθέναν άλλον στον κόσμο. Τον πιο δυνατό, έξυπνο και αξιόλογο άνθρωπο που γνώρισα ποτέ, που με παρότρυνε να σπουδάσω στο τμήμα των Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, που μου έμαθε να αγαπώ το έργο του μηχανικού και τον προγραμματισμό και μου έδειξε πώς να παλεύω για ό,τι αγαπώ. Ξωρίς αυτόν η ζωή μου δεν θα ήταν απλώς φτωχότερη, μα άδεια. Από την πρώτη μέρα στο πανεπιστήμιο, μέχρι και την τελευταία, ήταν το στήριγμα, η συμβουλή, η κριτική, η αγάπη και η φιλία που με κράτησαν όρθιο και με έκαναν να πιστεύω στον εαυτό μου. Ελπίζω να είμαι έστω και στο ελάχιστο τόσο καλός όσο είναι αυτός, και να τον κάνω περήφανο, όπως πάντοτε με έκανε ο ίδιος.

Τίτλος διπλωματικής

Θεοφάνης Θαρρόπουλος
theofact@ece.auth.gr

6 Οκτωβρίου 2024

Κατάλογος Σχημάτων

1.1 Παράδειγμα χρήσης του GitHub Copilot Chat εντός του IDE Visual Studio Code [88, 7]	6
1.2 Η αρχιτεκτονική του Language Server Protocol, Δανεισμένο από [73]	7
1.3 Παράδειγμα Ολοκλήρωσης Κώδικα εντός του IDE Visual Studio Code	8
1.4 Παράδειγμα Δημιουργίας Κώδικα σε πραγματικό χρόνο (real time code generation) εντός του IDE Visual Studio Code από το GitHub Copilot	9
1.5 Παράδειγμα Δημιουργίας Κώδικα σε πραγματικό χρόνο (real time code generation) εντός του IDE Visual Studio Code από το GitHub Copilot	10
2.1 Σχεσιακό Διάγραμμα Οντοτήτων της εφαρμογής	13
2.2 Δημοφιλέστερα Πλαίσια Διαδικτύου και τεχνολογίες του 2024, Δανεισμένο από [83]	15
2.3 Δημοφιλέστερες Βάσεις Δεδομένων του 2024, Δανεισμένο από [83]	16
2.4 Δημοφιλέστερες Γλώσσες Προγραμματισμού του 2024, Δανεισμένο από [83]	17
2.5 Δημοφιλέστερα περιβάλλοντα ανάπτυξης (IDE) του 2024, Δανεισμένο από [83]	18
2.6 Η μεθοδολογία δημιουργίας περσόνας για το μοντέλο, Δανεισμένο από [95]	20
2.7 Η προτροπή που δόθηκε στο μοντέλο για την δημιουργία της περσόνας	20
2.8 Παράδειγμα κώδικα που παρήχθη από το μοντέλο, με ορθογραφικό λάθος	21
2.9 Παράδειγμα διόρθωσης λάθους από το μοντέλο μεγάλης έκτασης, μετά την προτροπή του προγραμματιστή	22
2.10 Παράδειγμα διόρθωσης λάθους από το μοντέλο μικρής έκτασης, μετά την προτροπή του προγραμματιστή	22
2.11 Παράδειγμα διόρθωσης λάθους από τον προγραμματιστή, μετά την αδυναμία του μοντέλου να διορθώσει το λάθος	23
2.12 Παράδειγμα διόρθωσης λάθους μικρής έκτασης από τον προγραμματιστή, μετά την αδυναμία του μοντέλου να διορθώσει το λάθος	24
2.13 Συνολικά μετρικά ανά τύπο	25
2.14 Παράδειγμα μεθόδου που αφορά τα σχόλια, με πολλαπλές σχέσεις M:N	27
2.15 Σχεσιακό διάγραμμα της οντότητας του σχολίου	28
2.16 Τύπος δεδομένων για την δημιουργία της οντότητας της λίστας	29
2.17 Σχεσιακό διάγραμμα της οντότητας της λίστας	30
2.18 Παράδειγμα μεθόδου που αφορά την αλλαγή της σειράς των βιντεοπαιχνιδιών στη λίστα, Μέρος 1	31
2.19 Παράδειγμα μεθόδου που αφορά την αλλαγή της σειράς των βιντεοπαιχνιδιών στη λίστα, Μέρος 2	32
2.20 Παράδειγμα μεθόδου που αφορά την ανάκτηση λίστας με βάση το αναγνωριστικό της	33

2.21 Σχεσιακό διάγραμμα της σχέσης μεταξύ χρηστών και ακολούθων	34
2.22 Παράδειγμα μεθόδου που αφορά την δημιουργία ενός ελέγχου, με χρήση των μεθόδων <i>describe</i> και <i>it</i>	35
2.23 Παράδειγμα αποτελέσματος ελέγχου σε διεπαφή τερματικού, με χρήση των μεθόδων <i>describe</i> και <i>it</i> , μέσω του εργαλείου Jest	35
2.24 Παράδειγμα ελέγχου μονάδας για την επιτυχημένη δημιουργία παιχνιδιού	37
2.25 Παράδειγμα ελέγχου συμαβότητας για την επιτυχημένη δημιουργία παιχνιδιού	38
2.26 Ποσοστό αξιολόγησης απαντήσεων	40
2.27 Αριθμός αξιολόγησης απαντήσεων	40
2.28 Μέσος όρος συνεχόμενης αξιολόγησης απαντήσεων	41
2.29 Μέγιστη συνεχόμενη αξιολόγηση απαντήσεων	41
2.30 Αριθμός εμφανίσεων συνεχόμενων αξιολογήσεων άνω των τριών (3)	42
2.31 Αριθμός εμφανίσεων συνεχόμενων αξιολογήσεων άνω των δέκα (10)	42
2.32 Μέσος όρος αξιολόγησης απαντήσεων για θέματα, <i>μέρος 1</i>	43
2.33 Μέσος όρος αξιολόγησης απαντήσεων για θέματα, <i>μέρος 2</i>	43
2.34 Μέσος όρος αξιολόγησης απαντήσεων για θέματα, <i>μέρος 3</i>	44
3.1 Μέσος όρος αξιολόγησης απαντήσεων για θέματα, <i>μέρος 3</i>	49

Κατάλογος Πινάκων

2.1 Μικροαλλαγές ανά τύπο	24
2.2 Αλλαγές μεγάλης έκτασης ανά τύπο	24

Περιεχόμενα

1 Εισαγωγή	5
1.1 Δημιουργία Κώδικα (Code Generation) - Ολοκλήρωση Κώδικα (Code Completion)	6
2 Μεθοδολογία	11
2.1 Επιλογή Εφαρμογής	11
2.1.1 Τεχνολογική Στοίβα Technology Stack	14
2.2 Επιλογή Περιβάλλοντος Ανάπτυξης	17
2.3 Συλλογή Δεδομένων	18
2.3.1 Διαδικασία Συλλογής Δεδομένων	19
2.3.2 Ανάλυση Δεδομένων	24
2.4 Αξιολόγηση Απαντήσεων	39
2.5 Συμπεράσματα	44
3 Μοντέλο πρόβλεψης απόδοσης απαντήσεων βάσει προτροπής	46
3.1 Κατηγοριοποίηση των δεδομένων	46
3.1.1 Προεπεξεργασία και Προετοιμασία Δεδομένων	47
3.1.2 Επιλογή Χαρακτηριστικών	48
3.2 Εκπαίδευση και Αξιολόγηση Μοντέλων	48
3.3 Αποτελέσματα	48
Α' Ακρωνύμια και συντομογραφίες	50

Κεφάλαιο 1

Εισαγωγή

Η ανάπτυξη των Μεγάλων Γλωσσικών Μοντέλων (LLM) έχει επιφέρει ριζικές αλλαγές στον τομέα της Τεχνητής Νοημοσύνης και της Επεξεργασίας Φυσικής Γλώσσας (NLP) [16, 99, 100, 44]. Η εισαγωγή αυτών των μοντέλων στην καθημερινή ζωή μέσω του Chat-GPT το 2022 [63] έχει πυροδοτήσει μια επανάσταση στην τεχνολογική αγορά, σηματοδοτώντας την απαρχή του αγώνα για την κυριαρχία στην αγορά της Τεχνητής Νοημοσύνης [84, 54, 75, 50, 6]. Τα μοντέλα αυτά έχουν αποδειχθεί εξαιρετικά αποτελεσματικά στην αντιμετώπιση προβλημάτων επεξεργασίας φυσικής γλώσσας, όπως η αναγνώριση φυσικής γλώσσας, προάγοντας την ανάπτυξη της Γενικής Τεχνητής Νοημοσύνης (AGI) [8, 33].

Μέσα σε αυτή την επανάσταση, η χρήση μοντέλων επεξεργασίας φυσικής γλώσσας στον τομέα του προγραμματισμού και της ανάπτυξης λογισμικού, ονόματι βοηθοί κώδικα (Code Assistants), έχει αναδειχθεί ως ένας από τους πιο υποσχόμενους τομείς της τεχνολογίας. Ένα από τα πιο διαδεδομένα εργαλεία με αυτόν το σκοπό είναι το GitHub Copilot [32, 2]. Αναπτυγμένο σε συνεργασία με την OpenAI, το GitHub Copilot ξεκίνησε χρησιμοποιώντας το μοντέλο ονόματι Codex της OpenAI [21], σχεδιασμένο εξ αρχής αποκλειστικά για τη παραγωγή κώδικα, για να προτείνει κώδικα στον προγραμματιστή κατά την γραφή κώδικα.

Η παροδική απόσυρση του μοντέλου Codex τον Μάρτιο του 2023 και η οριστική του απόσυρση το 2023 [46], οδήγησε στην ανάπτυξη ενός νέου μοντέλου, σε συνεργασία μεταξύ της OpenAI, της Microsoft Azure AI, και της GitHub AI. Το νέο μοντέλο αρχικά βασίστηκε στο GPT-3.5 Turbo [98], με την επόμενή του έκδοση να βασίζεται στο GPT-4 [98], με το κωδικό όνομα GitHub Copilot X [25], δίνοντας την δυνατότητα για μια νέα λειτουργία, του GitHub Copilot Chat, ενός chatbot μοντέλου, παρόμοιο με αυτό του Chat-GPT. Μέσω αυτής της λειτουργίας, ο προγραμματιστής μπορεί μέσα από το περιβάλλον ανάπτυξής του (IDE), να κάνει ερωτήσεις στο μοντέλο, χρησιμοποιώντας φυσική γλώσσα, ενώ το μοντέλο μπορεί να χρησιμοποιήσει τον ήδη υπάρχοντα κώδικα, την τεκμηρίωση, και τις οδηγίες του προγραμματιστή για να απαντήσει στην ερώτηση του προγραμματιστή.

Σχήμα 1.1: Παράδειγμα χρήσης του GitHub Copilot Chat εντός του IDE Visual Studio Code [88, 7]

Το GitHub Copilot Chat αρχικά ήταν διαθέσιμο μόνο μέσω λίστας αναμονής, με την πρόσθαση να δίνεται σε περιορισμένο αριθμό προγραμματιστών, κατόπιν αίτησης, με την ενσωμάτωση του GPT-4 να γίνεται επίσημα τον Νοέμβριο του 2023 [79]. Η δημόσια κυκλοφορία του GitHub Copilot Chat έγινε τον Δεκέμβριο του 2023 [97].

Πέρα από το GitHub Copilot, υπάρχουν πολλοί άλλοι code assistants, με αυτούς που λήφθηκαν υπ' όψιν να είναι οι:

- Tabnine [90, 92]
- Codeium [19]
- Amazon Codewhisperer [15]

Η απόφαση για την χρήση του GitHub Copilot ως το εργαλείο, του οποίου η λειτουργία θα εξεταστεί στην παρούσα διπλωματική, έγινε με βάση την ευρεία χρήση του, την ενσωμάτωση του μοντέλου GPT-4, και την δυνατότητα χρήσης του GitHub Copilot Chat, καθώς και την δωρεάν παροχή του σε ενεργούς φοιτητές του Αριστοτελείου Πανεπιστήμιου Θεσσαλονίκης μέσω του προγράμματος GitHub Student Developer Pack [3].

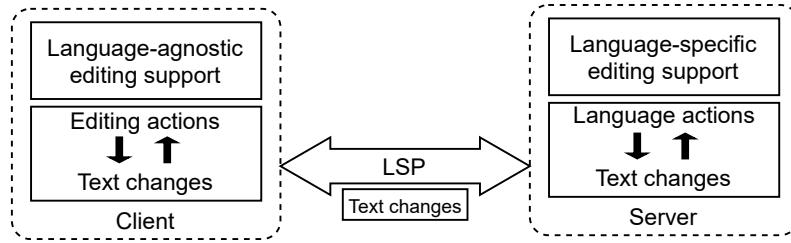
1.1 Δημιουργία Κώδικα (Code Generation) - Ολοκλήρωση Κώδικα (Code Completion)

Δύο όροι που συχνά συναντώνται στην βιβλιογραφία είναι η **Δημιουργία Κώδικα** (Code Generation) και η **Ολοκλήρωση Κώδικα** (Code Completion). Και οι δύο όροι έχουν άμεση σχέση με την παραγωγικότητα του προγραμματιστή και αποτελούν σημαντικά

εργαλεία για την ανάπτυξη λογισμικού [52, 49, 12], και ενώ και τα δύο μπορούν να χρησιμοποιήσουν μοντέλα Τεχνητής Νοημοσύνης [86, 72] με την διαφορά να βρίσκεται στον τρόπο λειτουργίας τους.

Ολοκλήρωση Κώδικα

Σύμφωνα με Omar et al. [62], η Ολοκλήρωση Κώδικα αφορά εργαλεία που τα περισσότερα περιβάλλοντα ανάπτυξης παρέχουν μέσω της μορφής πλωτού μενού που περιέχει συμφραζόμενες-σχετικές μεταβλητές, πεδία, μεθόδους, τύπους και άλλα αποσπάσματα κώδικα. Επιλέγοντας από το μενού, οι προγραμματιστές μπορούν να αποφύγουν πολλά συνηθισμένα ορθογραφικά και λογικά λάθη και να εξαλείψουν τις περιπτές πληκτρολογήσεις. Το βασικό εργαλείο που χρησιμοποιείται για την επίτευξη του σκοπού αυτού είναι το Language Server Protocol (LSP) της Microsoft, δίνοντας την δυνατότητα σε κάθε περιβάλλον ανάπτυξης να επικοινωνήσει με την εξωτερική διεργασία του Language Server και να λάβει πληροφορίες για τον κώδικα που γράφεται, όπως τις προτάσεις ολοκλήρωσης κώδικα, τα λάθη, και τις προτάσεις διόρθωσης, σε μια διάταξη χρήστη - διακομιστή [71, 17].



Σχήμα 1.2: Η αρχιτεκτονική του Language Server Protocol, Δανεισμένο από [73]

Η Ολοκλήρωση Κώδικα δεν παράγει κώδικα από το μηδέν, αλλά προτείνει συμπληρωματικές λύσεις στον υπάρχοντα κώδικα, βοηθώντας τον προγραμματιστή να ολοκληρώσει τον κώδικα του ταχύτερα.

The screenshot shows a code editor window in Visual Studio Code. The code being typed is:

```
return; },  
),  
getById: pu .input(z. .query(as  
.const r .find .whe  
.i [es] OpenInNewWindowIcon  
, [es] Open_Sans  
) [es] Option  
then((res) ->
```

A code completion dropdown menu is open, listing several options:

- OkImpl
- Old_Standard_...
- Oldenburg
- Ole
- Oleo_Script
- Oleo_Script_Swash_C...
- Onest
- Oooh_Baby
- OpacityIcon
- OpenInNewWindowIcon
- Open_Sans
- Option

The dropdown has a header "ts-results" and a "lucide-react" section at the bottom.

Σχήμα 1.3: Παράδειγμα Ολοκλήρωσης Κώδικα εντός του IDE Visual Studio Code

Δημιουργία Κώδικα

Η Δημιουργία Κώδικα αφορά την άμεση παραγωγή κώδικα από ένα μοντέλο στοχασχτικά, με βάση το συγκείμενο κώδικα του προγραμματιστή. Το μοντέλο που χρησιμοποιείται για την παραγωγή κώδικα είναι εκπαιδευμένο σε μεγάλα σύνολα δεδομένων και μπορεί να παράγει κώδικα από το μηδέν. Το μοντέλο αυτό μπορεί να παράγει κώδικα σε πολλές γλώσσες προγραμματισμού, όπως Python, JavaScript, C++ κ.α. Η παραγωγή κώδικα μπορεί να γίνει μέσω μιας διεπιφάνεις (API) που παρέχεται από το μοντέλο, ή μέσω μιας επέκτασης ενός ειδικού περιβάλλοντος ανάπτυξης, όπως στην περίπτωση του GitHub Copilot.

src > server > api > routers > **TS** game.ts > [e] gameRouter > **get**

```
16  export interface GameDetails extends Omit<Game, "franchiseId" | "publisherId"> {
11    platforms: Array<{
8      >;
7      franchise: Omit<Franchise, "image">;
6      publisher: Omit<Publisher, "image">;
5      reviews?: Array<ReviewContext>;
4    }
3
2  export const gameRouter = createTRPCRouter({
1    get getAll: publicProcedure.query()
53   | async (): Promise<Result<Array<Game>, TRPCError>> => {
      |   const result: Result<Array<Game>, TRPCError> = await ctx.prisma.game
      |     .findMany()
      |     .then((res) => Ok(res), handlePrismaError);
      |
      |   return result;
    }
1
2
3  getById: publicProcedure
4    .input(z.object({ id: z.string().cuid2() }))
5    .query(async ({ ctx, input }): Promise<Result<Game, TRPCError>> => {
6      const result: Result<Game, TRPCError> = await ctx.prisma.game
7        .findOne({
8          where: {
9            id: input.id,
  
```

Σχήμα 1.4: Παράδειγμα Δημιουργίας Κώδικα σε πραγματικό χρόνο (real time code generation) εντός του IDE Visual Studio Code από το GitHub Copilot

Η παραγωγή κώδικα γίνεται με διάφορους βαθμούς λεπτομέρειας. Αρχικά, το μοντέλο υπολογίζει το επόμενο σύμβολο του κώδικα χρησιμοποιώντας **next token prediction** [39, 47, 93, 27, 23, 22]. Στο στάδιο της ολοκλήρωσης ολόκληρης σειράς κώδικα, το μοντέλο χρησιμοποιεί ένα κομμάτι συγκείμενου κώδικα. [39, 37, 85, 51]. Στο μέγιστο δυνατό βαθμό, το μοντέλο μπορεί να παράγει ολόκληρες συναρτήσεις ή κλάσεις. [29, 37, 2].

Η παραγωγή κώδικα επίσης μπορεί να λάβει υπ' όψιν αποκλειστικά το συγκείμενο κώδικα πριν την θέση του κέρσορα στο αρχείο ή και τον κώδικα που ακολουθεί την θέση του κέρσορα. [40] Το μοντέλο του GitHub Copilot χρησιμοποιεί την δεύτερη προσέγγιση, παράγοντας κώδικα που συμπληρώνει τον υπάρχοντα κώδικα του προγραμματιστή. [2, 29, 94]. Κατά την έρευνα λήφθηκε η απόφαση να χρησιμοποιηθεί, ως επί το πλείστον, η λειτουργία GitHub Copilot Chat για την έρευνα και τα πειράματα, κυρίως γιατί ο σύγκειμενος κώδικας που χρησιμοποιεί το μοντέλο κατά την λειτουργία αυτή, μπορεί να επιλεχθεί είτε από το μοντέλο, είτε συγκεκριμένα από τον προγραμματιστή. Ένας ακόμα λόγος που επιλέχθηκε η χήση του GitHub Copilot Chat, είναι ότι η καταγραφή του κώδικα που παράχθηκε κατά την έρευνα από το μοντέλο ήταν πολύ δυσκολότερη με την χρήση του GitHub Copilot, καθώς η δημιουργία κώδικα σε πραγματικό χρόνο (**real time code generation**) ήταν πολύ δύσκολη αυτής του GitHub Copilot Chat.

The screenshot shows a Visual Studio Code interface with the following details:

- File:** game.ts
- Content:**

```

1  export interface GameDetails extends Omit<Game, "franchiseId" | "publisherId"> {
2    _count: { reviews: number };
3    developers: Array<Omit<Developer, "image">};
4    features: Array<Omit<Feature, "image">};
5    genres: Array<Genre>;
6    platforms: Array<{
7      storeLink: string;
8      platform: Omit<Platform, "image">;
9    }>;
10   franchise: Omit<Franchise, "image">;
11   publisher: Omit<Publisher, "image">;
12   reviews: Array<ReviewContext>;
13 }
14
15 export const gameRouter = createTRPCRouter({
16   getAll: publicProcedure.query(
17     async ({ ctx }: { ctx: any }): Promise<Result<Array<Game>, TRPCError>> => {
18       const result: Result<Array<Game>, TRPCError> = await ctx.prisma.game
19         .findMany()
20         .then((res) => Ok(res), handlePrismaError);
21       return result;
22     }
23   ),
24   getById: publicProcedure
25     .input(z.object({ id: z.string().cuid2() }))
26     .query(async ({ ctx, input }: { ctx: any; input: { id: string } }): Promise<Result<Game, TRPCError>> => {
27       const result: Result<Game, TRPCError> = await ctx.prisma.game
28         .findUnique({
29           where: {
30             id: input.id,
31           },
32         })
33     }
34   ),
35 });
36
37 // You, 6 months ago + feat(trpc): game router
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
79
80
81
82
83
84
85
86
87
88
89
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
109
110
111
112
113
114
115
116
117
118
119
119
120
121
122
123
124
125
126
127
128
129
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
149
150
151
152
153
154
155
156
157
158
159
159
160
161
162
163
164
165
166
167
168
169
169
170
171
172
173
174
175
176
177
178
179
179
180
181
182
183
184
185
186
187
188
189
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
209
210
211
212
213
214
215
216
217
218
219
219
220
221
222
223
224
225
226
227
227
228
229
229
230
231
232
233
234
235
236
237
237
238
239
239
240
241
242
243
244
245
245
246
247
247
248
249
249
250
251
252
253
254
255
255
256
257
257
258
259
259
260
261
262
263
264
264
265
266
266
267
267
268
268
269
269
270
270
271
271
272
272
273
273
274
274
275
275
276
276
277
277
278
278
279
279
280
280
281
281
282
282
283
283
284
284
285
285
286
286
287
287
288
288
289
289
290
290
291
291
292
292
293
293
294
294
295
295
296
296
297
297
298
298
299
299
300
300
301
301
302
302
303
303
304
304
305
305
306
306
307
307
308
308
309
309
310
310
311
311
312
312
313
313
314
314
315
315
316
316
317
317
318
318
319
319
320
320
321
321
322
322
323
323
324
324
325
325
326
326
327
327
328
328
329
329
330
330
331
331
332
332
333
333
334
334
335
335
336
336
337
337
338
338
339
339
340
340
341
341
342
342
343
343
344
344
345
345
346
346
347
347
348
348
349
349
350
350
351
351
352
352
353
353
354
354
355
355
356
356
357
357
358
358
359
359
360
360
361
361
362
362
363
363
364
364
365
365
366
366
367
367
368
368
369
369
370
370
371
371
372
372
373
373
374
374
375
375
376
376
377
377
378
378
379
379
380
380
381
381
382
382
383
383
384
384
385
385
386
386
387
387
388
388
389
389
390
390
391
391
392
392
393
393
394
394
395
395
396
396
397
397
398
398
399
399
400
400
401
401
402
402
403
403
404
404
405
405
406
406
407
407
408
408
409
409
410
410
411
411
412
412
413
413
414
414
415
415
416
416
417
417
418
418
419
419
420
420
421
421
422
422
423
423
424
424
425
425
426
426
427
427
428
428
429
429
430
430
431
431
432
432
433
433
434
434
435
435
436
436
437
437
438
438
439
439
440
440
441
441
442
442
443
443
444
444
445
445
446
446
447
447
448
448
449
449
450
450
451
451
452
452
453
453
454
454
455
455
456
456
457
457
458
458
459
459
460
460
461
461
462
462
463
463
464
464
465
465
466
466
467
467
468
468
469
469
470
470
471
471
472
472
473
473
474
474
475
475
476
476
477
477
478
478
479
479
480
480
481
481
482
482
483
483
484
484
485
485
486
486
487
487
488
488
489
489
490
490
491
491
492
492
493
493
494
494
495
495
496
496
497
497
498
498
499
499
500
500
501
501
502
502
503
503
504
504
505
505
506
506
507
507
508
508
509
509
510
510
511
511
512
512
513
513
514
514
515
515
516
516
517
517
518
518
519
519
520
520
521
521
522
522
523
523
524
524
525
525
526
526
527
527
528
528
529
529
530
530
531
531
532
532
533
533
534
534
535
535
536
536
537
537
538
538
539
539
540
540
541
541
542
542
543
543
544
544
545
545
546
546
547
547
548
548
549
549
550
550
551
551
552
552
553
553
554
554
555
555
556
556
557
557
558
558
559
559
560
560
561
561
562
562
563
563
564
564
565
565
566
566
567
567
568
568
569
569
570
570
571
571
572
572
573
573
574
574
575
575
576
576
577
577
578
578
579
579
580
580
581
581
582
582
583
583
584
584
585
585
586
586
587
587
588
588
589
589
590
590
591
591
592
592
593
593
594
594
595
595
596
596
597
597
598
598
599
599
600
600
601
601
602
602
603
603
604
604
605
605
606
606
607
607
608
608
609
609
610
610
611
611
612
612
613
613
614
614
615
615
616
616
617
617
618
618
619
619
620
620
621
621
622
622
623
623
624
624
625
625
626
626
627
627
628
628
629
629
630
630
631
631
632
632
633
633
634
634
635
635
636
636
637
637
638
638
639
639
640
640
641
641
642
642
643
643
644
644
645
645
646
646
647
647
648
648
649
649
650
650
651
651
652
652
653
653
654
654
655
655
656
656
657
657
658
658
659
659
660
660
661
661
662
662
663
663
664
664
665
665
666
666
667
667
668
668
669
669
670
670
671
671
672
672
673
673
674
674
675
675
676
676
677
677
678
678
679
679
680
680
681
681
682
682
683
683
684
684
685
685
686
686
687
687
688
688
689
689
690
690
691
691
692
692
693
693
694
694
695
695
696
696
697
697
698
698
699
699
700
700
701
701
702
702
703
703
704
704
705
705
706
706
707
707
708
708
709
709
710
710
711
711
712
712
713
713
714
714
715
715
716
716
717
717
718
718
719
719
720
720
721
721
722
722
723
723
724
724
725
725
726
726
727
727
728
728
729
729
730
730
731
731
732
732
733
733
734
734
735
735
736
736
737
737
738
738
739
739
740
740
741
741
742
742
743
743
744
744
745
745
746
746
747
747
748
748
749
749
750
750
751
751
752
752
753
753
754
754
755
755
756
756
757
757
758
758
759
759
760
760
761
761
762
762
763
763
764
764
765
765
766
766
767
767
768
768
769
769
770
770
771
771
772
772
773
773
774
774
775
775
776
776
777
777
778
778
779
779
780
780
781
781
782
782
783
783
784
784
785
785
786
786
787
787
788
788
789
789
790
790
791
791
792
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
800
801
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
```

Κεφάλαιο 2

Μεθοδολογία

Στο παρόν κεφαλαίο αυτό παρουσιάζεται η μεθοδολογία που ακολουθήθηκε κατά την διάρκεια της σύλλογης των δεδομένων των πειραμάτων με το GitHub Copilot και με το GitHub Copilot Chat. Αρχικά, η επιλογή της εφαρμογής πάνω στην οποία το GitHub Copilot αξιολογήθηκε, περιγράφεται παρακάτω.

2.1 Επιλογή Εφαρμογής

Για την επιλογή της κατάλληλης εφαρμογής, λήφθηκαν οι παρακάτω παράμετροι υπ' όψιν:

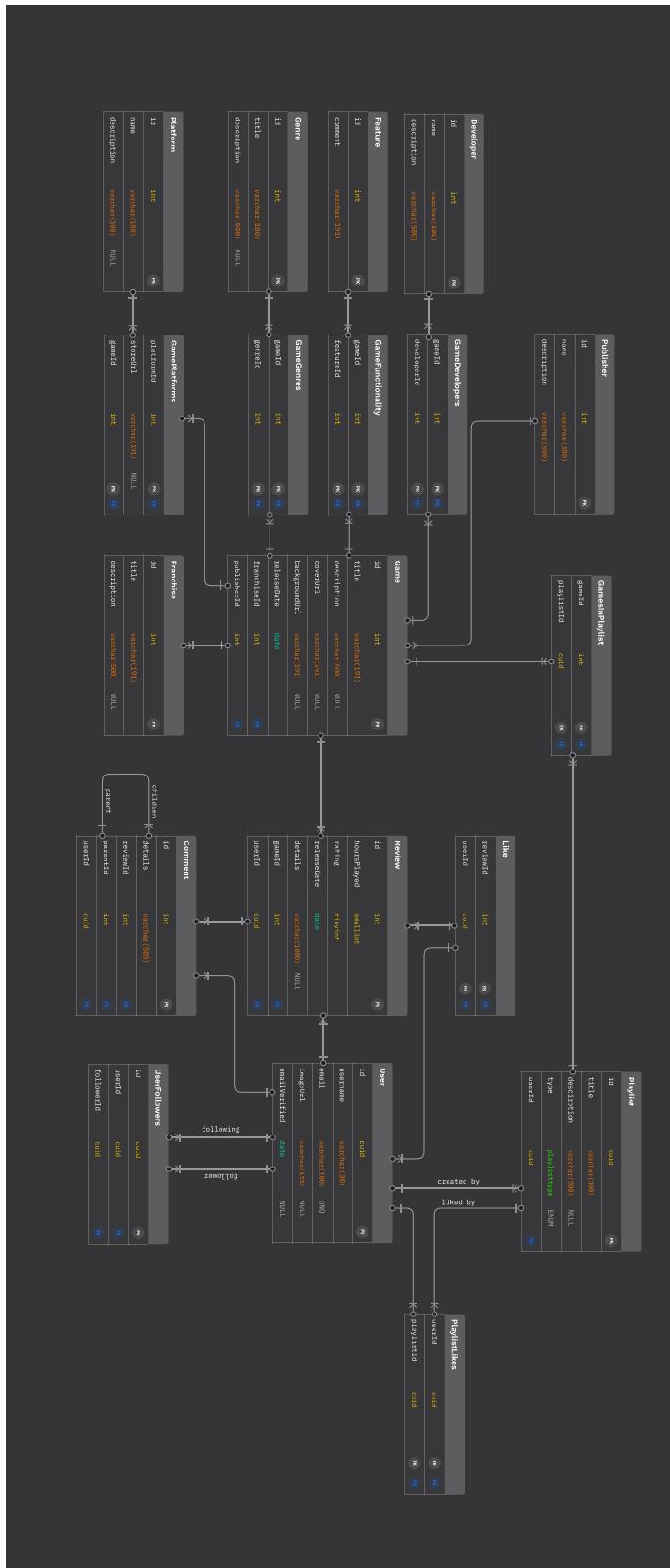
- **Πολυπλοκότητα της εφαρμογής:** Η εφαρμογή πρέπει να είναι αρκετά πολύπλοκη ώστε να απαιτεί την χρήση του GitHub Copilot για την ανάπτυξη του κώδικα.
- **Τύπος της εφαρμογής:** Η εφαρμογή πρέπει να είναι μια ιδέα αντίστοιχη με τις υπόλοιπες υλοποιήσεις του κλάδου της Μηχανικής Λογισμικού και συγκεκριμένα της Ανάπτυξης Ιστότοπων **Web Development**.
- **Γλώσσα της εφαρμογής και Τεχνολογική Στοίβα (Technology Stack):** Η επιλογή της Τεχνολογικής Στοίβας πρέπει να γίνει με βάση μοντέρνες τεχνολογίες και με ευρεία χρήση, προκειμένου το μοντέλο να έχει την περισσότερη εμπειρία και τις καλύτερες αποδόσεις, αλλά και να γίνεται μια αξιολόγηση με πραγματικά και εξελισσόμενα εργαλεία.

Με βάση τις παραπάνω παραμέτρους, επιλέχθηκε η ανάπτυξη μιας εφαρμογής η οποία θα αποτελείται από έναν ιστότοπο που θα παρέχει ένα μέσο δικτύωσης που αφορά τα βιντεοπαιχνίδια. Η εφαρμογή αυτή είχε ήδη αναπτυχθεί στα πλαίσια του μαθήματος 'Μηχανική Λογισμικού Ι', επομένως οι βασικές ανάγκες και λειτουργίες της εφαρμογής είχαν καταγραφθεί. Η επιλογή αυτή έγινε προκειμένου η αξιολόγηση του κώδικα και των προτάσεων του GitHub Copilot να μην έχει αλλαγές με βάση τις αλλαγές που προέκυψαν κατά τον σχεδιασμό της εφαρμογής. Η εφαρμογή αποτελείται από τρεις (3) βασικές λειτουργίες:

- Κριτική παιχνιδιών και κοινοποίηση των κριτικών των χρηστών με τους υπόλοιπους χρήστες της εφαρμογής.

- Διαχείριση και κοινοποίηση λιστών (playlists) από βιντεοπαιχνίδια με τους υπόλοιπους χρήστες της εφαρμογής.
- Δυνατότητα στον χρήστη να σχολιάσει τις κριτικές των άλλων χρηστών, καθώς και να ‘ακολουθήσει’ άλλους χρήστες, λαμβάνοντας ειδοποιήσεις για τις ενέργειές τους.

Παρατίθεται το Σχεσιακό Διάγραμμα Οντοτήτων της εφαρμογής:



Σχήμα 2.1: Σχεσιακό Διάγραμμα Οντοτήτων της εφαρμογής

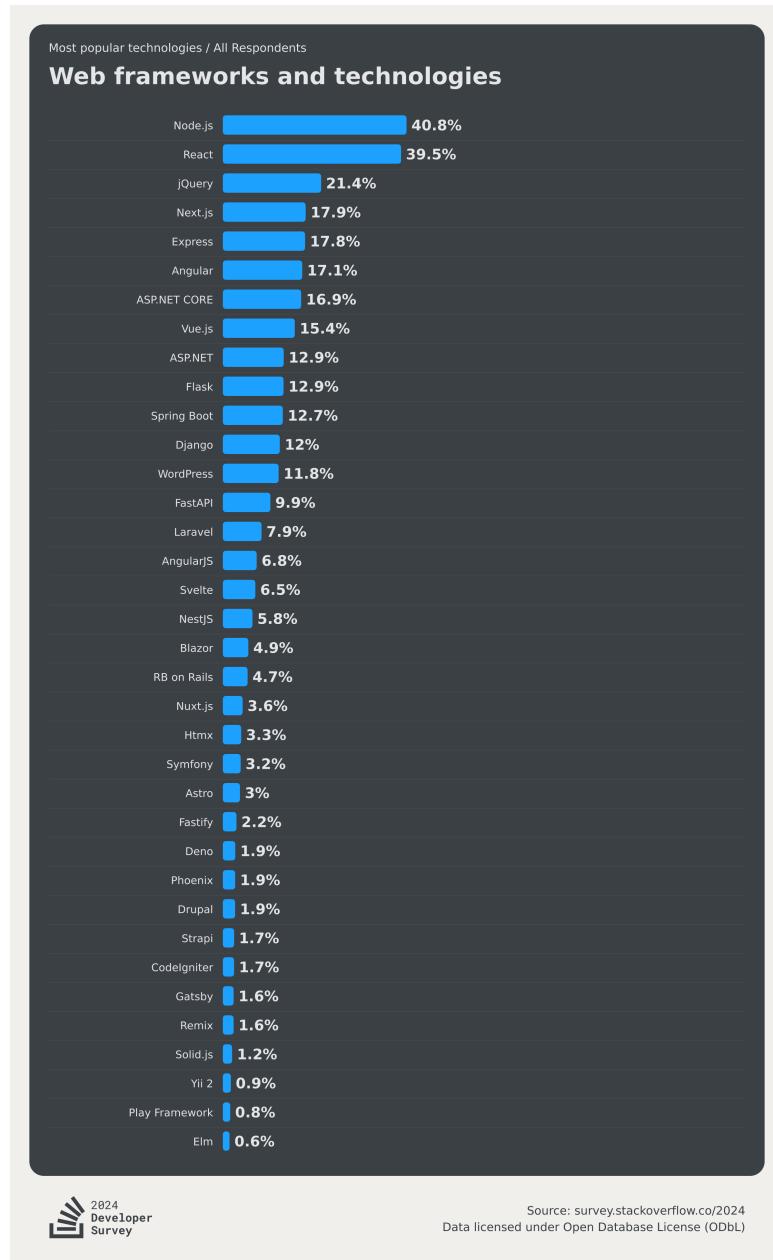
2.1.1 Τεχνολογική Στοίβα Technology Stack

Η επιλογή της Τεχνολογικής Στοίβας έγινε με βάση την ευρεία χρήση των τεχνολογιών αυτών, την ευκολία στην ανάπτυξη και την ευκολία στην ενσωμάτωση του GitHub Copilot. Η εφαρμογή αναπτύχθηκε με την χρήση των παρακάτω τεχνολογιών:

- **Γλώσσα Προγραμματισμού:** Η εφαρμογή αναπτύχθηκε με την χρήση της γλώσσας προγραμματισμού Typescript [55], ενός συντακτικού υπερσυνόλου της γλώσσας JavaScript [60], μια από τις πιο διαδομένες γλώσσες προγραμματισμού στον κόσμο. [18, 80] Συγκεκριμένα, το 2024 η Javascript ήταν η πρώτη στην λίστα της επισκόπησης του Stack Overflow, μετά από 60.171 ψήφους. [83] Την πρώτη θέση επίσης έλαβε το 2023 και το 2022. [81, 82]

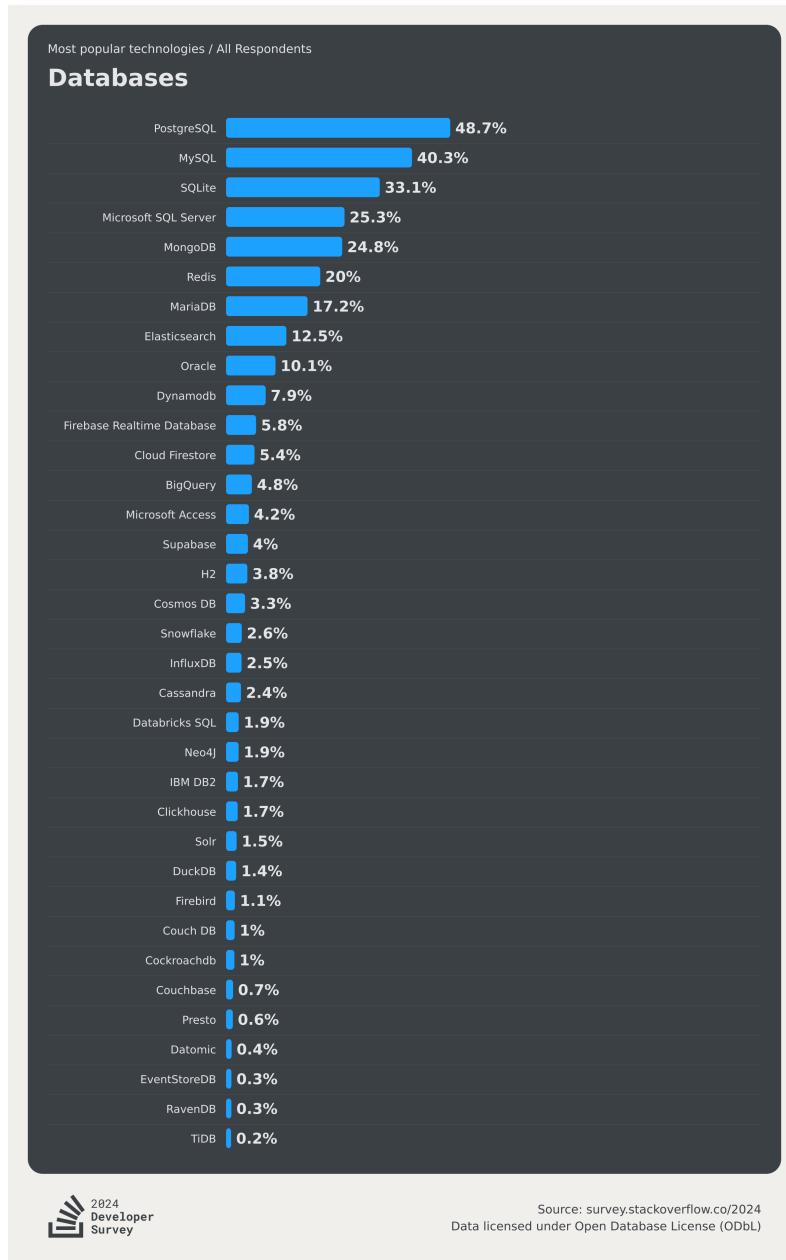
Η γλώσσα Javascript επίσης, είναι η γλώσσα προγραμματισμού που εκτελείται στις ιστοσελίδες του διαδικτύου, μέσω της μηχανής V8 της Google [35] για τους περιηγητές διαδικτύου που βασίζονται στην μηχανή περιηγητή ανοιχτού κώδικα Chromium [34], της μηχανής SpiderMonkey της Mozilla [58] για τους περιηγητές που χρησιμοποιούν τη μηχανή περιηγητή ανοιχτού κώδικα Gecko [57], και της μηχανής JavaScriptCore της Apple [10] για τους περιηγητές που χρησιμοποιούν τη μηχανή περιηγητή WebKit [11], δίνοντάς της τον τίτλο της γλώσσας του διαδικτύου.

- **Πίσω Ανάπτυξη (Backend Development):** Η πίσω ανάπτυξη της εφαρμογής έγινε με την χρήση της βιβλιοθήκης tRPC [89], μιας βιβλιοθήκης που παρέχει την δυνατότητα δημιουργίας API με την χρήση της γλώσσας Typescript και του περιβάλλοντος εκτέλεσης (runtime) Node.js [65]. Το Node.js βρισκόταν στην πρώτη θέση για πλαίσιο διαδικτύου (web framework) στην επισκόπηση του Stack Overflow το 2024 [83].



Σχήμα 2.2: Δημοφιλέστερα Πλαίσια Διαδικτύου και τεχνολογίες του 2024, Δανεισμένο από [83]

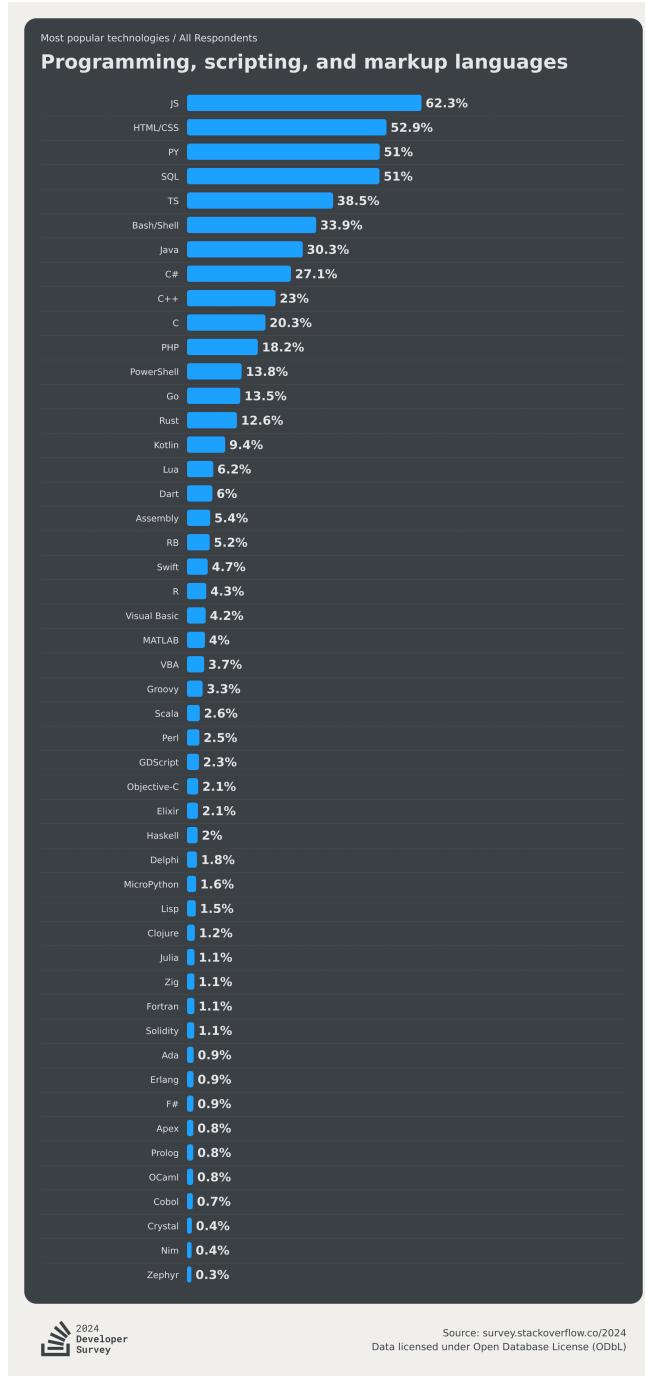
Τα δεδομένα της εφαρμογής αποθηκεύτηκαν σε μια βάση δεδομένων MySQL [66], μια από τις πιο διαδεδομένες σχεσιακές βάσεις δεδομένων [83]. Το εργαλείο της Αντικειμενο-σχεσιακής Απεικόνισης (ORM) που χρησιμοποιήθηκε ήταν το Prisma [69], το οποίο παρέχει την δυνατότητα δημιουργίας απλών και ασφαλών ερωτημάτων (queries) στην βάση δεδομένων. Το σύστημα διαχείρισης της ταυτότητας των χρηστών (authentication) και των δικαιωμάτων τους (authorization) υλοποιήθηκε με την χρήση της βιβλιοθήκης NextAuth.js [61].



Σχήμα 2.3: Δημοφιλέστερες Βάσεις Δεδομένων του 2024, Δανεισμένο από [83]

- **Εμπρόσθια Ανάπτυξη (Frontend Development):** Η εμπρόσθια ανάπτυξη της εφαρμογής έγινε με την χρήση της React [53], μιας βιβλιοθήκης της JavaScript για την ανάπτυξη διεπαφών χρήστη, την πλέον πιο ευρέως γνωστή βιβλιοθήκη Javascript για ανάπτυξη διεπαφών χρήστη [83]. Συγκεκριμένα, χρησιμοποιήθηκε το πλαίσιο εργασίας Next.js [91], το οποίο παρέχει δυνατότητες όπως την προφόρτωση των σελίδων, την δυνατότητα δημιουργίας στατικών ιστοσελίδων, και την δυνατότητα δημιουργίας δυναμικών ιστοσελίδων, αποτελώντας ένα από τα πιο ευρέως χρησιμοποιημένα πλαίσια εργασίας.
- **Έλεγχος Λογισμικού (Software Testing):** Για τον έλεγχο της λειτουργίας του παραγόμενου κώδικα, χρησιμοποιήθηκε το πλαίσιο εργασίας Jest [64], το οποίο παρέχει την δυνατότητα δημιουργίας και εκτέλεσης δοκιμαστικών συνόλων κώδικα, με σκοπό την εξασφάλιση της άρτιας λειτουργίας του κώδικα. [41, 38, 5, 56,

Η παραπάνω στοίβα ονομάστηκε T3 stack, έχοντας πλέον δημιουργήσει μια μεγάλη κοινότητα στον κλάδο της ανάπτυξης ιστότοπων και της ανάπτυξης λογισμικού. [87]

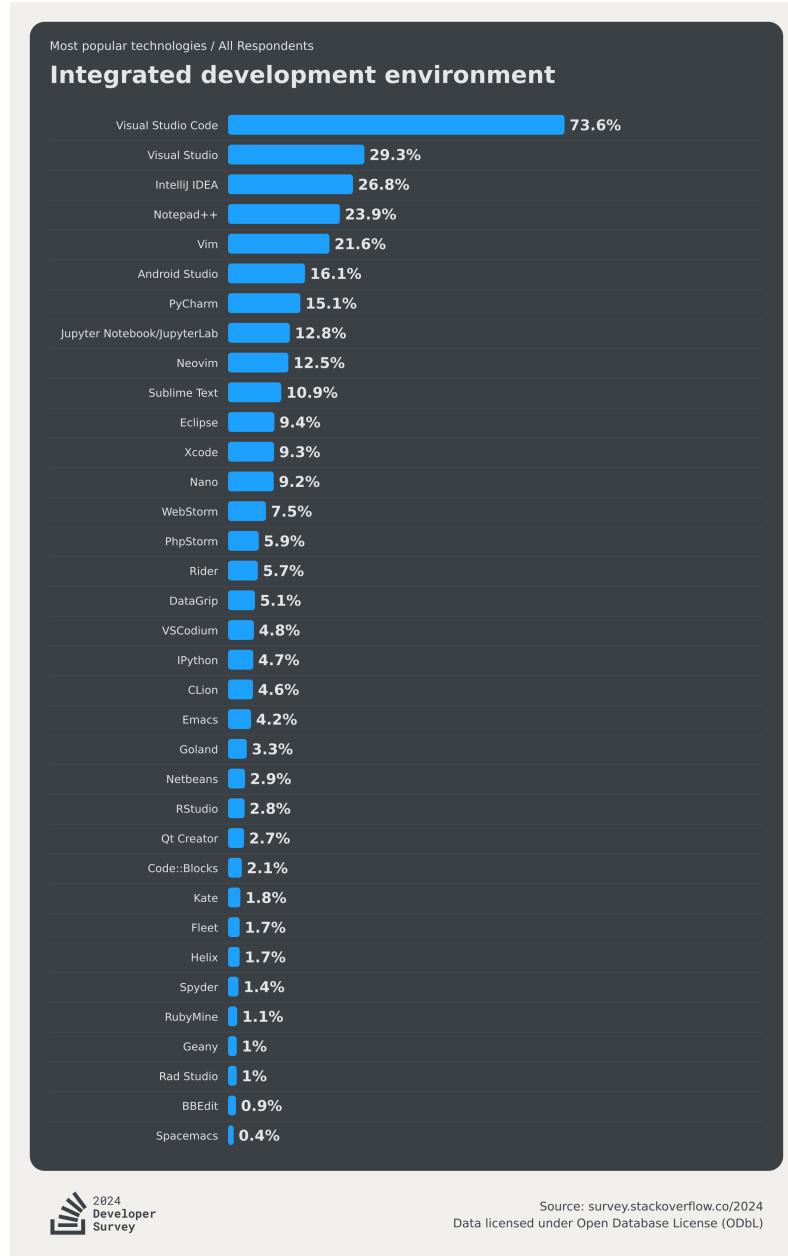


Σχήμα 2.4: Δημοφιλέστερες Γλώσσες Προγραμματισμού του 2024, Δανεισμένο από [83]

2.2 Επιλογή Περιβάλλοντος Ανάπτυξης

Η επιλογή του περιβάλλοντος ανάπτυξης έγινε με βάση την ευκολία στην ενσωμάτωση του GitHub Copilot και την ευρεία χρήση του περιβάλλοντος ανάπτυξης. Η επιλογή έγινε στο Visual Studio Code [7], το πιο δημοφιλές, σύμφωνα με την επισκόπηση του

Stack Overflow του 2024, περιθάλλον ανάπτυξης. [83]. Το Visual Studio Code είναι ένα περιθάλλον ανάπτυξης ανεπιγμένο από την Microsoft το 2015, χρησιμοποιώντας την ανοιχτού κώδικα πλατφόρμα Code - OSS. Το GitHub Copilot Chat ήταν για πρώτη φορά διαθέσιμο στο Visual Studio Code μέσω της ομώνυμης επέκτασης. [31]



Σχήμα 2.5: Δημοφιλέστερα περιθάλλοντα ανάπτυξης (IDE) του 2024, Δανεισμένο από [83]

2.3 Συλλογή Δεδομένων

Για την συλλογή των δεδομένων, αρχικά δημιουργήθηκε ο σκελετός της βάσης κώδικα (codebase), καθώς και μια λειτουργικότητα της εφαρμογής, μαζί με τα αντίστοιχα αρχεία ελέγχου της λειτουργίας του κώδικα. Η λειτουργικότητα που επιλέχθηκε να αναπτυχθεί χωρίς την χρήση του μοντέλου ήταν αυτής της σειράς του βιντεοπαιχνιδιού

(Franchise). Η επιλογή έγινε γιατί η λειτουργικότητα αυτή ήταν αρκετά απλή, καθώς η σειρά έχει μόνο σχέσεις 1:M με οντότητα του βιντεοπαιχνιδιού (Game) ::.

Με αυτό τον τρόπο, δημιουργήθηκαν μέθοδοι για την δημιουργία, την ανάγνωση, την ενημέρωση και την διαγραφή των σειρών του βιντεοπαιχνιδιού (CRUD), καθώς και έλεγχοι για την σωστή εκτέλεση αυτών, με βάση την ταυτοποίηση και των δικαιωμάτων των χρηστών. Σκοπός της προεργασίας αυτής ήταν η αξιολόγηση του κατά πόσο το μοντέλο θα ακολουθούσε τις κατευθυντήριες γραμμές που ορίστηκαν από τον προγραμματιστή.

2.3.1 Διαδικασία Συλλογής Δεδομένων

Για την καλύτερη μορφή και κατηγοριοποίηση των δεδομένων, η κάθε προτροπή χωρίστηκε σε τέσσερις (4) κατηγορίες, με βάση το είδος της:

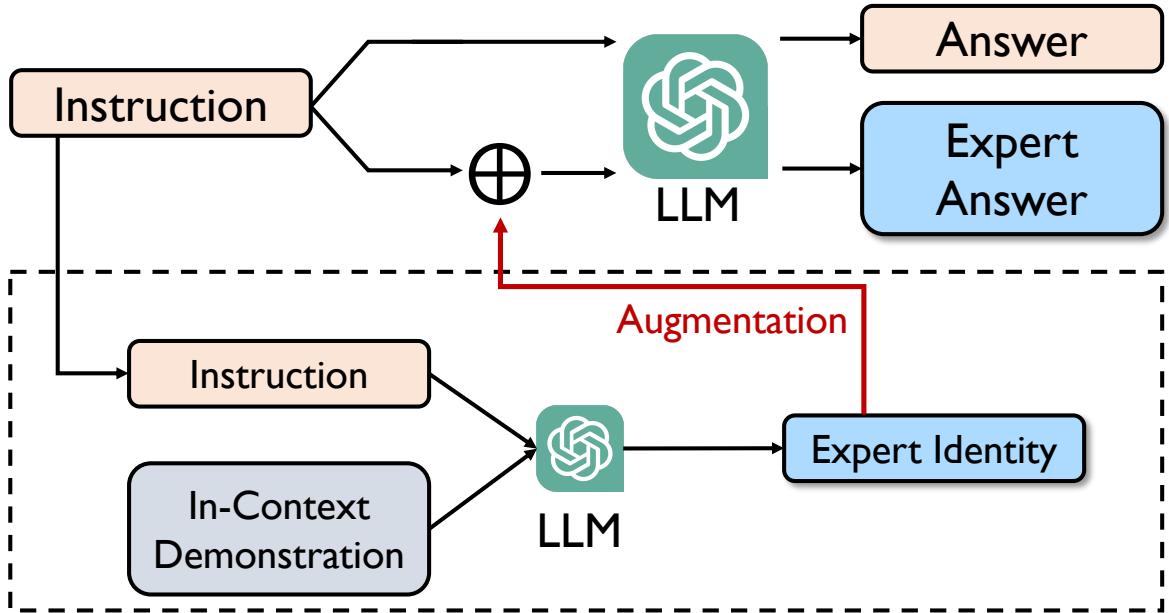
- **Γλώσσας (Language):** Προτροπές που αφορούν ερωτήσεις για την γλώσσα προγραμματισμού (συντακτικό, δομή).
- **Πίσω Ανάπτυξης (Backend):** Προτροπές που αφορούν ερωτήσεις για την ανάπτυξη του API της εφαρμογής.
- **Ελέγχου (Testing):** Προτροπές που αφορούν ερωτήσεις για την ανάπτυξη ελέγχου για τις μεθόδους που έχουν γραφτεί.
- **Άλλες:** Προτροπές που αφορούν ερωτήσεις που δεν σχετίζονται με την συγγραφή κώδικα (για την λειτουργικότητα του GitHub Copilot, για την δομή των απαντήσεών του, τυχόν λάθη από πλευρά του προγραμματιστή κατά την σύνταξη της προτροπής).

Πέρα από την αρχική κατηγοριοποίηση των προτροπών, η κάθε προτροπή επίσης κατηγοριοποίηθηκε περαιτέρω, με το κάθε θέμα της ερώτησης, όπως για παράδειγμα ανάπτυξη των μεθόδων που αφορούσαν μια οντότητα της εφαρμογής και ανάπτυξη ελέγχων αυτών των μεθόδων, εκ των οπίσιων ελέγχων χωρίστηκαν σε Integration Tests (έλεγχοι συμβατότητας) και σε Unit Tests (έλεγχοι μονάδας) [42, 45]. Οι διαφορές μεταξύ των δύο είναι οι εξής:

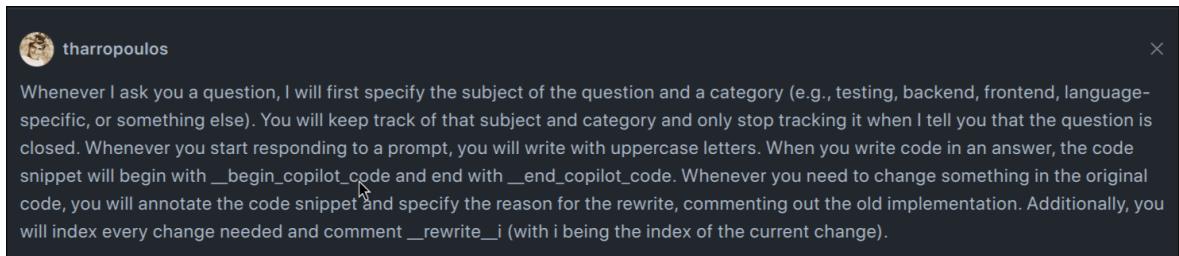
- **Unit Testing (Έλεγχος Μονάδας):** Το Unit Testing επικεντρώνεται στον έλεγχο μεμονωμένων κομματιών κώδικα, συνήθως μεθόδων ή συναρτήσεων, σε απομόνωση. Στοχεύει στην επαλήθευση ότι κάθε μονάδα του λογισμικού λειτουργεί σωστά.
- **Integration Testing (Έλεγχος Συμβατότητας):** Το Integration Testing εξετάζει πώς διαφορετικά μέρη του συστήματος λειτουργούν μαζί. Ελέγχει την αλληλεπίδραση μεταξύ διαφορετικών μονάδων ή συστατικών για να διασφαλίσει ότι συνεργάζονται σωστά ως σύνολο. Η ειδοποιός διαφορά είναι ότι το Unit Testing εστιάζει σε μεμονωμένα κομμάτια κώδικα, ενώ το Integration Testing εξετάζει πώς αυτά τα κομμάτια συνεργάζονται. [68]

Για την συλλογή των δεδομένων, δημιουργήθηκε μια περσόνα ενός προγραμματιστή για το μοντέλο [101, 9, 95], με τους κανόνες να ακολουθεί κάθε του απάντηση μια συγκεκριμένη δομή. Ζητήθηκε από το μοντέλο να αρχίζει κάθε μήνυμα με το θέμα της ερώτησης, τον τύπο της ερώτησης και έναν άριθμο που αναπαριστά την σειρά της ερώτησης για το συγκεκριμένο θέμα. Στη συνέχεια, ζητήθηκε από το μοντέλο να βάζει σχόλια στον κώδικα που πάραγε, υποδεικνύοντας την αρχή και το τέλος του παραγόμενου

κώδικα, με τη μορφή `BEGIN_COPILOT_CODE / END_COPILOT_CODE`. Τέλος, ζητάθηκε από το μοντέλο να παράγει κώδικα που να ακολουθεί τις κατευθυντήριες γραμμές που ορίστηκαν από τον προγραμματιστή.



Σχήμα 2.6: Η μεθοδολογία δημιουργίας περσόνας για το μοντέλο, Δανεισμένο από [95]



Σχήμα 2.7: Η προτροπή που δόθηκε στο μοντέλο για την δημιουργία της περσόνας

Η επιλογή να ζητηθεί από το μοντέλο να υποδεικνύει τον κώδικα που παράγει έγινε προκειμένου να συλλεχθούν δεδομένα για την ποιότητα του κώδικα που παρήχθη, αλλά και για να καταγραφτούν οι φορές που ο προγραμματιστής επενέβη, με σκοπό να διορθώσει λάθη που το μοντέλο αδυνατούσε. Η τακτική αυτή προϋπόθετε ότι ο προγραμματιστής και χειριστής του μοντέλου θα είχε την δυνατότητα να αντιληφθεί τα λάθη που παρήχθησαν από το μοντέλο, και να τα διορθώσει, καθώς και να γνωρίζει πώς να προτρέψει το μοντέλο να απαντήσει την επιθυμητή απάντηση. Χωρίς αυτήν την απόφαση, το μοντέλο θα παρήγαγε κώδικα ο οποίος δεν θα ήταν λειτουργικός, καθώς υπήρχαν συντακτικά λάθη που δεν μπορούσε να παρατηρήσει.



```
1 // BEGIN_COPILOT_CODE
2 describe("and the user is following the creator", () => {
3   it("should return the playlist", async () => {
4     // Arrange
5     const playlist = await prisma.playlist.create({
6       data: {
7         name: faker.lorem.words(3),
8         description: faker.lorem.words(),
9         type: "CUSTOM",
10        visibility: "FOLLOWERS_ONLY",
11        user: {
12          connect: { id: mockAdmin.id },
13        },
14      },
15    });
16    ...
17 // END_COPILOT_CODE
```

Σχήμα 2.8: Παράδειγμα κώδικα που παρήχθη από το μοντέλο, με ορθογραφικό λάθος

Η διόρθωση λαθών από τον προγραμματιστή αποτέλεσε έσχατη λύση, καθώς πρώτη επιλογή ήταν η αποφυγή συγγραφής κώδικα από τον προγραμματιστή, μέσω επισύμανσης των λαθών στο μοντέλο, με την προσδοκία ότι το μοντέλο θα διόρθωνε το λάθος χωρίς την ανάγκη επέμβασης του προγραμματιστή. Τα λάθη που διορθώθηκαν από το μοντέλο μετά την προτροπή του προγραμματιστή σημειεύθηκαν με σχόλια με πρόθημα *REWRITE*, για αλλαγές σε μία σειρά κώδικα, ενώ με σχόλια με πρόθημα *REVISION* για αλλαγές σε μικρά κομμάτια κώδικα.

```
● ● ●  
1 data: {  
2   // REVISION_1: Correctly connect the user and game to the review  
3   // content: input.content,  
4   // rating: input.rating,  
5   // userId: ctx.user.id,  
6   // userId: ctx.session.user.id,  
7   // gameId: input.gameId,  
8   ...rest,  
9   createdAt: new Date(),  
10  updatedAt: new Date(),  
11  user: {  
12    connect: {  
13      id: ctx.session.user.id,  
14    },  
15  },  
16  game: {  
17    connect: {  
18      id: gameId,  
19    },  
20  },  
21  ...
```

Σχήμα 2.9: Παράδειγμα διόρθωσης λάθους από το μοντέλο μεγάλης έκτασης, μετά την προτροπή του προγραμματιστή

```
● ● ●  
1 // REWRITE_1: remove deprecated image field  
2 //   image: faker.image.imageUrl(),  
3 image: faker.image.url(),
```

Σχήμα 2.10: Παράδειγμα διόρθωσης λάθους από το μοντέλο μικρής έκτασης, μετά την προτροπή του προγραμματιστή

Κατά την διόρθωση λαθών, ο προγραμματιστής άφηνε σχόλια, εξηγώντας το λόγο που παρενέβη, με σκοπό την καλύτερη δυνατή τεκμηρίωση, αλλά και την εκμάθηση του μοντέλου, με σκοπό την αποφυγή μελλοντικών λαθών, σημειώνοντας τα κομμάτια κάθικα με τα σχόλια *BEGIN_NON_COPILOT_CODE / END_NON_COPILOT_CODE*.

```

1 // BEGIN_NON_COPILOT_CODE
2 // Explicit prisma relations are hard to work with, copilot struggles with them
3 getGames: publicProcedure
4   .input(z.object({ id: z.string().cuid2() }))
5   .query(
6     async ({
7       ctx,
8       input,
9     }): Promise<
10    Result<Array<GameToPlatform & { game: Game }>, TRPCError>
11  > => {
12    const result: Result<
13      Array<GameToPlatform & { game: Game }>,
14      TRPCError
15    > = await ctx.prisma.platform
16      .findUnique({
17        where: {
18          id: input.id,
19        },
20        select: {
21          games: {
22            include: {
23              game: true,
24            },
25          },
26        },
27      })
28    ...
29 // END_NON_COPILOT_CODE

```

Σχήμα 2.11: Παράδειγμα διόρθωσης λάθους από τον προγραμματιστή, μετά την αδυναμία του μοντέλου να διορθώσει το λάθος

Λάθη που διορθώθηκαν μέσω μιας γραμμής (όπως ονομασίες μεταβλητών ή συμβόλων), σημειώθηκαν με σχόλια με πρόθημα *MANUAL_REWRITE*. Επίσης, κατά την χειροκίνητη συγγραφή κώδικα, το μοντέλο του GitHub Copilot, έδινε στον προγραμματιστή προτάσεις διόρθωσης του κώδικα. Ορισμένες προτάσεις που έλυναν το πρόβλημα, ενσωματώθηκαν στον τελικό κώδικα, σημειώνοντας την αρχή και το τέλος του κώδικα που παρήχθη σε πραγματικό χρόνο, μέσω σχολίων *BEGIN_COPILOT_SUGGESTION / END_COPILOT_SUGGESTION*, σημειώνοντας την αρχή και το τέλος του προτεινόμενου κομματιού κώδικα.



```
1 // MANUAL_REWRITE: correct rating type
2 // rating: faker.datatype.number({ min: 1, max: 5 }),
3 rating: faker.number.int({ min: 1, max: 5 }),
```

Σχήμα 2.12: Παράδειγμα διόρθωσης λάθους μικρής έκτασης από τον προγραμματιστή, μετά την αδυναμία του μοντέλου να διορθώσει το λάθος

2.3.2 Ανάλυση Δεδομένων

Μετά την συλλογή των δεδομένων, ακολούθησε η ανάλυσή των ευρημάτων, μέσω ενός αναλυτή (lexer [14]), προκειμένου να μετρηθούν τα εξής:

- Ο αριθμός των περιστατικών επέμβασης του προγραμματιστή σε μεγάλες εκτάσεις κώδικα
- Ο αριθμός των περιστατικών επέμβασης του προγραμματιστή σε μικρές εκτάσεις κώδικα
- Ο αριθμός των περιστατικών που το μοντέλο διόρθωσε το λάθος σε μικρές εκτάσεις κώδικα
- Ο αριθμός των περιστατικών που το μοντέλο διόρθωσε μεγαλύτερα λάθη, έπειτα από προτροπή του προγραμματιστή

Τύπος	Μικροαλλαγές Copilot	Μικροαλλαγές προγραμματιστή
Backend	20	3
Integration Tests	26	7
Unit Tests	32	3

Πίνακας 2.1: Μικροαλλαγές ανά τύπο

Τύπος	Αλλαγές Copilot	Αλλαγές προγραμματιστή
Backend	3	10
Integration Tests	30	14
Unit Tests	4	3

Πίνακας 2.2: Αλλαγές μεγάλης έκτασης ανά τύπο

Τα δεδομένα λήφθηκαν μέσα από εικοσιοκτώ (28) αρχεία και πεντακόσιες εικοσιπέντε (525) προτροπές και συγκεκριμένα:

- Δεκαεπτά (17) αρχεία που αφορούν τον έλεγχο του κώδικα
- Έντεκα (11) αρχεία που αφορούν την ανάπτυξη του API

Παρατίθεται ένα διάγραμμα που αναπαριστά τα αποτελέσματα της ανάλυσης:



Σχήμα 2.13: Συνολικά μετρικά ανά τύπο

Όπως φαίνεται από το διάγραμμα ::, οι περισσότερες αλλαγές που πραγματοποιήθηκαν από την πλευρά του μοντέλου έγιναν σε μικρές εκτάσεις κώδικα, ενώ οι περισσότερες αλλαγές που πραγματοποιήθηκαν από την πλευρά του προγραμματιστή έγιναν σε μεγάλες εκτάσεις κώδικα. Σε κάθε τύπο αρχείου παρατηρείται αυτή η τάση, με στον προγραμματιστή να χρειάζεται να επεμβεί πολύ σπανιότερα σε σχέση με το μοντέλο και συγκεκριμένα, σε κάθε τύπο αρχείου, οι επεμβάσεις του προγραμματιστή για μικρές εκτάσεις κώδικα είναι οι λιγότερες.

Αξιοσημείωτη επίσης η παρατήρηση ότι στην περίπτωση των αρχειών που αφορούσαν το Unit Testing, οι αλλαγές σε μικρή έκταση κώδικα από το μοντέλο ήταν με διαφορά οι περισσότερες, ενώ παράλληλα οι υπόλοιπες αλλαγές ήταν ελάχιστες. Αυτό μπορεί να οφείλεται στη χρήση βιβλιοθηκών, με βάση τον έλεγχο του κώδικα και πιο συγκεκριμένα αυτή της faker.js[1]. Η συγκεκριμένη βιβλιοθήκη αφορά την δημιουργία ψεύτικων δεδομένων, στο ίδιο πνεύμα με το Lorem Ipsum [4], αλλά για δεδομένα που αφορούν ένα λογισμικό, όπως ημερομηνίες, ονόματα, διευθύνσεις ηλεκτρονικού ταχυδρομίου κ.α.. Η βιβλιοθήκη έχει λάβει πολλές εκδόσεις τα τελευταία χρόνια, με αλλαγές στον τρόπο που καλούνται οι μέθοδοι του API της, με αποτέλεσμα το μοντέλο να μην μπορεί να ακολουθήσει πάντοτε τις αλλαγές αυτές. Σημαντικό να σημειωθεί επίσης ότι το μοντέλο αδυνατούσε να παρατηρήσει τις αλλαγές που ζητήθηκαν από τον προγραμματιστή, με αποτέλεσμα να ζητηθεί μέσω διαφορετικής προτροπής η αλλαγή της ίδιας κλήσης μιας μεθόδου πάνω από μία φορά στο ίδιο αρχείο.

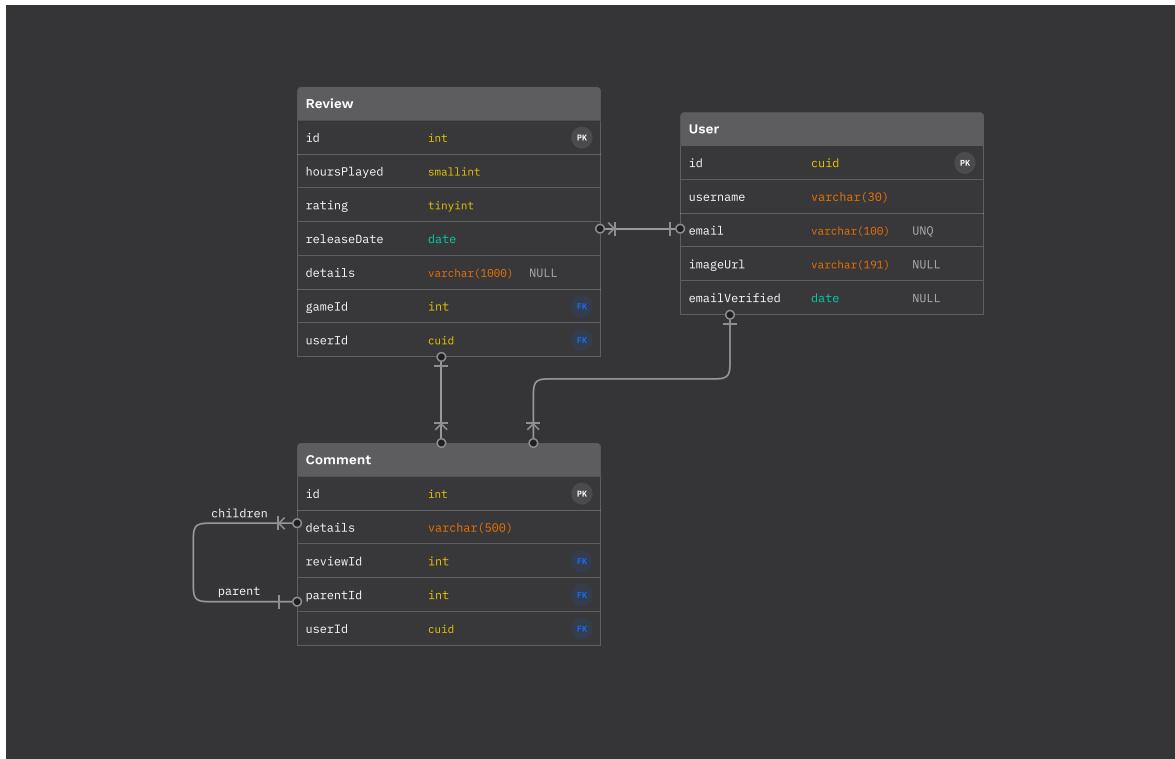
Στην περίπτωση των αρχείων που αφορούσαν το Backend, οι απαιτούμενες αλλαγές ήταν οι λίγοτερες σε πλήθος, με τη διαφορά ότι οι περισσότερες επεμβάσεις του προγραμματιστή αφορούσαν την γραφή ολόκληρων μεθόδων. Η απουσία των απαραίτητων αλλαγών, οφείλεται στο γεγονός ότι η βασική λογική πίσω από ένα μεγάλο ποσοστό των μεθόδων αφορά την γραφή βασικών CRUD μεθόδων, οι οποίες είναι προκαθορισμένες και δεν απαιτούν αλλαγές. Αυτό έχει ως αποτέλεσμα το μοντέλο να μην χρειάζεται να προτρέπεται να αλλάξει την λογική των μεθόδων, αλλά μόνο την υλοποίηση τους. Στις περιπτώσεις μεθόδων που επηρέαζαν οντότητες με σχέσεις M:N [70], το μοντέλο του GitHub Copilot αδυνατούσε να ακολουθήσει την λογική, με αποτέλεσμα να χρειαστεί

να υλοποιηθούν από τον προγραμματιστή.

Παράδειγμα αποτελεί η περίπτωση της οντότητας του σχολίου. Τα σχόλια πραγματοποιούνται σε κριτικές των χρηστών, παρομοίως όπως σε άλλες εφαρμογές κοινωνικών δικτύων. Τα σχόλια πραγματοποιούνται σε κριτικές των χρηστών, παρομοίως όπως σε άλλες εφαρμογές κοινωνικών δικτύων. Κάθε σχόλιο έχει μια σχέση 1:M με την κριτική, μια σχέση 1:M με τον χρήστη, μια σχέση M:1 με τις αντιδράσεις τύπου Mou αρέσει (Likes), αλλά μπορεί να έχει και άλλα σχόλια που συνδέονται με αυτό, δημιουργώντας ένα νήμα (thread) σχολίων, παρόμοιο με έναδιατεταγμένο Κ-οστο δέντρο [30]. Η λογική πίσω από την υλοποίηση των μεθόδων που αφορούν τα σχόλια, είναι περίπλοκη, με τα ερωτήματα που απαιτούνται να σταλθούν προς στη βάση μέσω του Prisma να είναι πολλά και περίπλοκα. Το μοντέλο του GitHub Copilot αδυνατούσε να ακολουθήσει την λογική, με αποτέλεσμα να χρειαστεί να υλοποιηθούν από τον προγραμματιστή.

```
1 const baseSelectClause = Prisma.validator<Prisma.CommentSelect>()({  
2   _count: {  
3     select: {  
4       likes: true,  
5       children: false,  
6     },  
7   },  
8   id: true,  
9   content: true,  
10  user: {  
11    select: {  
12      image: true,  
13      id: true,  
14      name: true,  
15    },  
16  },  
17  reviewId: true,  
18  createdAt: true,  
19  updatedAt: true,  
20});  
21  
22 const parentSelectClause = {  
23   ...baseSelectClause,  
24   // Don't take the children (this is where the current comment is)  
25   children: false,  
26   // Don't take the parent (this is two depths above)  
27   parent: false,  
28 };  
29  
30 const childrenSelectClause = {  
31   // Don't take the parent (current comment)  
32   ...baseSelectClause,  
33   parent: false,  
34   // Take the first depth of children  
35   children: {  
36     // Take the second depth of children, and no more than that  
37     select: parentSelectClause,  
38   },  
39};
```

Σχήμα 2.14: Παράδειγμα μεθόδου που αφορά τα σχόλια, με πολλαπλές σχέσεις M:N



Σχήμα 2.15: Σχεσιακό διάγραμμα της οντότητας του σχολίου

Στη περίπτωση της οντότητας της λίστας (Playlist), η λογική της λειτουργικότητας αυτής είναι να δίνει στο χρήστη τη δυνατότητα να δημιουργεί λίστες με τα αγαπημένα του βιντεοπαιχνίδια. Ο κάθε χρήστης έχει κατά την εγγραφή του στην εφαρμογή πέντε (5) λίστες:

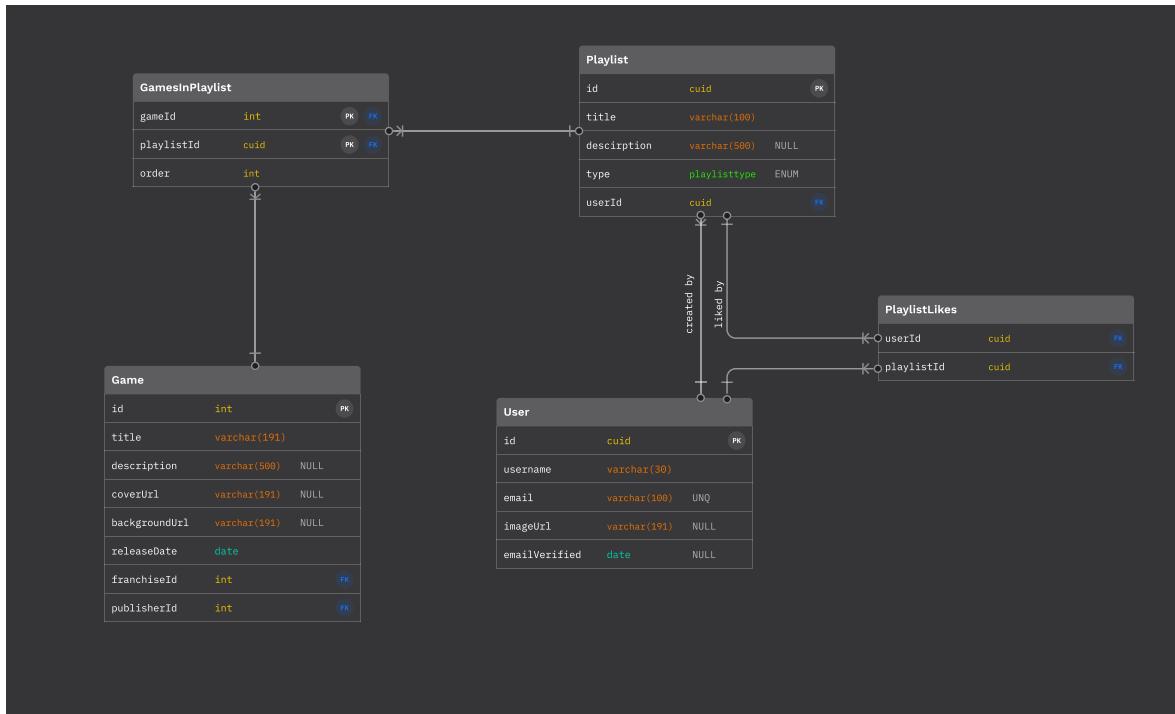
- Αγαπημένα (Liked): Τα βιντεοπαιχνίδια που άρεσαν στον χρήστη, η πιο εύχρηστη λίστα για τον χρήστη
- Λίστα αναμονής (Backlog): Τα βιντεοπαιχνίδια που θέλει να παίξει στο μέλλον
- Ολοκληρωμένα (Completed): Τα βιντεοπαιχνίδια που έχει παίξει και φτάσει στο τέλος τους στο παρελθόν παίξει στο μέλλον
- Ενεργά (Playing): Τα βιντεοπαιχνίδια που παίζει αυτή τη χρονική περίοδο
- Εγκαταλειμμένα (Dropped): Τα βιντεοπαιχνίδια που δεν του άρεσαν και τα έχει εγκαταλείψει

Ο χρήστης έχει το δικαίωμα να δημιουργήσει όσες ακόμα προσαρμοσμένες (Custom) λίστες θέλει, και να τις μοιραστεί είτε με τους ακολούθους του, είτε να τις κάνει δημόσιες, είτε να τις κρατήσει ως ιδιωτικές.

```
● ● ●  
1 export const createPlaylistSchema = z.object({  
2   name: z.string(),  
3   description: z.string().optional(),  
4   type: z.enum([  
5     "BACKLOG",  
6     "LIKED",  
7     "COMPLETED",  
8     "PLAYING",  
9     "DROPPED",  
10    "CUSTOM",  
11  ]),  
12  visibility: z.enum(["PUBLIC", "PRIVATE", "FOLLOWERS_ONLY"]),  
13});
```

Σχήμα 2.16: Τύπος δεδομένων για την δημιουργία της οντότητας της λίστας

Κάθε λίστα έχει μια σχέση 1:M με τον χρήστη, μια σχέση 1:M με τις αντιδράσεις, και μία σχέση N:M με βιντεοπαιχνίδια. Η σχέση με τα βιντεοπαιχνίδια υλοποιείται μέσω ενός πίνακα σύνδεσης [13], ο οποίος, πέρα από τα ξένα κλειδιά του βιντεοπαιχνιδιού και της λίστας, περιέχει και το πεδίο order, το οποίο καθορίζει τη σειρά με την οποία τα βιντεοπαιχνίδια εμφανίζονται στη λίστα. Η λογική πίσω από την υλοποίηση προϋπόθεται τη χρήση μιας διπλά συνδεμένης λίστας, προκειμένου να διατηρείται η επιθυμητή σειρά, αν ο χρήστης αποφασίσει να αλλάξει τη σειρά με την οποία τα βιντεοπαιχνίδια θα εμφανίζονται στη λίστα.



Σχήμα 2.17: Σχεσιακό διάγραμμα της οντότητας της λίστας

Το μοντέλο του GitHub Copilot αδυνατούσε να ακολουθήσει την λογική, με αποτέλεσμα να χρειαστεί να υλοποιηθούν εξ ολοκλήρου από τον προγραμματιστή.

```

1 updateOrder: protectedProcedure
2   .input(
3     z.object({
4       id: z.string().cuid2(),
5       gameId: z.string().cuid2(),
6       order: z.number().int(),
7     })
8   )
9   .mutation(
10    async ({{
11      ctx,
12      input,
13    }}): Promise<Result<Playlist & { games: Array<GameToPlaylist & { game: Game }> }, TRPCError>> => {
14      const whereClause = Prisma.validator<Prisma.PlaylistWhereInput>()({
15        id: input.id,
16        deleted: null,
17        user: { id: ctx.session.user.id },
18      });
19
20      const oldOrder = await ctx.prisma.gameToPlaylist
21        .findUnique({
22          where: {
23            gameId_playlistId: {
24              gameId: input.gameId,
25              playlistId: input.id,
26            },
27          },
28          select: { order: true },
29        })
30      .then((res) => Ok(res), handlePrismaError);
31
32      if (!oldOrder.ok) return oldOrder;
33
34      const increment = ctx.prisma.playlist.update({
35        where: whereClause,
36        data: {
37          games: {
38            updateMany: {
39              where: {
40                order: {
41                  gte: input.order,
42                },
43              },
44              data: {
45                order: {
46                  increment: 1,
47                },
48              },
49            },
50          },
51        },
52      });
53

```

Σχήμα 2.18: Παράδειγμα μεθόδου που αφορά την αλλαγή της σειράς των βιντεοπαιχνιδιών στη λίστα, *Μέρος 1*

```

1  const decrement = ctx.prisma.playlist.update({
2    where: whereClause,
3    data: {
4      games: {
5        updateMany: {
6          where: {
7            order: {
8              lte: input.order,
9            },
10           },
11           data: {
12             order: {
13               decrement: 1,
14             },
15           },
16           },
17         },
18       },
19     });
20
21 const update = ctx.prisma.playlist.update({
22   where: whereClause,
23   data: {
24     games: {
25       update: {
26         where: {
27           gameId_playlistId: {
28             gameId: input.gameId,
29             playlistId: input.id,
30           },
31         },
32         data: {
33           order: input.order,
34         },
35       },
36     },
37   },
38   include: {
39     games: {
40       orderBy: {
41         order: "asc",
42       },
43       include: {
44         game: true,
45       },
46     },
47   },
48 });
49
50 switch (true) {
51   case oldOrder instanceof Ok && oldOrder.val instanceof Object && typeof oldOrder.val.order === "number":
52     const result: Result<Playlist & { games: Array<GameToPlaylist & { game: Game }>} , TRPCError> =
53       oldOrder.val.order > input.order
54         ? await ctx.prisma.$transaction([increment, update]).then(([res] => Ok(res[1]), handlePrismaError)
55         : await ctx.prisma.$transaction([decrement, update]).then(([res] => Ok(res[1]), handlePrismaError));
56     return result;
57   default:
58     return new Err(new TRPCError({ code: "INTERNAL_SERVER_ERROR" }));
59   }
60 }
61 ),

```

Σχήμα 2.19: Παράδειγμα μεθόδου που αφορά την αλλαγή της σειράς των βιντεοπαιχνιδιών στη λίστα, *Μέρος 2*

Πέρα από τη σειρά των βιντεοπαιχνιδιών, σημαντική ήταν και η εξασφάλιση της ορθής ορατότητας της λίστας σε άλλους χρήστες, χωρίς να επιτρέπεται η πρόσθιαση σε λίστες που δεν είναι είτε δημόσιες, είτε ανήκουν σε χρήστες που ο χρήστης ακολουθεί, είτε είναι προσωπικές λίστες του ίδιου. Στην περίπτωση αυτή, αν και το μοντέλο ακολούθησε τη σωστή λογική, αδυνάτισε να βρει το σωστό ερώτημα προς τη βάση όσο αφορά την ορατότητα της λίστας προς τους ακολούθους του χρήστη.

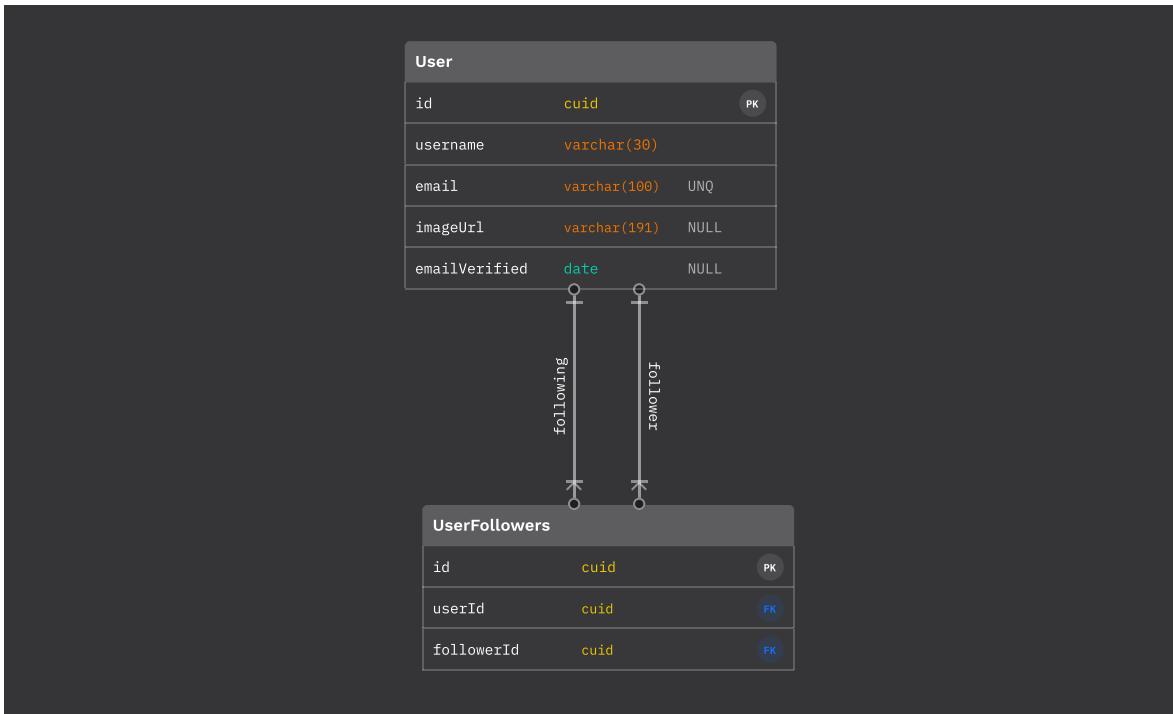
```

1  getById: publicProcedure
2    .input(z.object({ id: z.string().cuid2() }))
3    .query(async ({ ctx, input }): Promise<Result<Playlist, TRPCError>> => {
4      const whereClause: Prisma.PlaylistWhereInput = {
5        id: input.id,
6        deleted: null,
7        OR: [{ visibility: "PUBLIC" }],
8      };
9
10     if (ctx.session?.user?.id) {
11       (whereClause.OR as Prisma.PlaylistWhereInput[]).push(
12         // REWRITE_2: use user relation instead of userId
13         // {userId: ctx.session.user.id},
14         { user: { id: ctx.session.user.id } },
15         {
16           visibility: "FOLLOWERS_ONLY",
17           // END_COPILOT_CODE
18           // BEGIN_NON_COPILOT_CODE
19           // Copilot didn't find the bug
20           // user: { followers: { some: { id: ctx.session.user.id } } },
21           user: {
22             followers: { some: { follower: { id: ctx.session.user.id } } },
23           },
24           // END_NON_COPILOT_CODE
25         }
26       );
27     }
28
29     const result: Result<Playlist, TRPCError> = await ctx.prisma.playlist
30       // REWRITE_1: use findfirst instead of findUnique
31       // .findUnique({ where: whereClause })
32       .findFirst({ where: whereClause })
33       .then((res) => {
34         return res
35         ? Ok(res)
36         : new Err(
37           new TRPCError({
38             code: "NOT_FOUND",
39             message: "Playlist not found",
40           })
41         );
42       }, handlePrismaError);
43
44     return result;
45   },

```

Σχήμα 2.20: Παράδειγμα μεθόδου που αφορά την ανάκτηση λίστας με βάση το αναγνωριστικό της

Όπως παρατηρείται στο διάγραμμα ;;, το μοντέλο ορθά αναγνώρισε ότι σε περίπτωση που μια λίστα δημοσιεύονταν μόνο στους ακολούθους του δημιουργού της, θα έπρεπε ο χρήστης που την ανακτά θα έπρεπε να ανήκει στην λίστα των ακολούθων του δημιουργού, η υλοποίηση του κώδικα δεν ήταν σωστή, καθώς αγνόησε την M:N σχέση μεταξύ των χρηστών και των ακολούθων τους.



Σχήμα 2.21: Σχεσιακό διάγραμμα της σχέσης μεταξύ χρηστών και ακολούθων

Στις περιπτώσεις των αρχείων που αφορούσαν το Integration Testing, οι αλλαγές ήταν οι περίσσοτερες συγκριτικά με τις περιπτώσεις ανάπτυξης του API και του Unit Testing και οι αλλαγές από την πλευρά του μοντέλου ήταν αρκετά πιο ισορροπημένες, με τις αλλαγές σε μεγάλη έκταση κώδικα να υπερβαίνουν αυτών της μικρής έκτασης κώδικα.

Στην περίπτωση αυτή, ρόλο έπαιξε το γεγονός ότι κατά την γραφή των ελέγχων, το εργαλείο Jest, παρέχει τη δυνατότητα στον προγραμματιστή να συντάξει μια περιγραφή για τον έλεγχο, μέσω των μεθόδων `describe` και `it`. Οι μέθοδοι αυτές, παίρνουν ως πρώτο όρισμα μια συμβολοσειρά που περιγράφει τον έλεγχο, και ως δεύτερο όρισμα μια συνάρτηση `callback` [24] που περιέχει τον κώδικα του ελέγχου. Η περιγραφή του ελέγχου, ακολουθεί την ανάπτυξη **Λογισμικού Καθοδηγούμενη Από Τη Συμπεριφορά** (Behavior Driven Development, BDD).

Η BDD είναι μια προσέγγιση στην ανάπτυξη λογισμικού που επικεντρώνεται στη συμπεριφορά του συστήματος από την οπτική γωνία του χρήστη. Ενθαρρύνει τη συνεργασία μεταξύ προγραμματιστών, ελεγκτών και μη τεχνικών ενδιαφερομένων, χρησιμοποιώντας μια κοινή γλώσσα για να περιγράψει την επιθυμητή συμπεριφορά του λογισμικού. Συνήθως περιλαμβάνει τη συγγραφή σεναρίων ή παραδειγμάτων που περιγράφουν πώς το λογισμικό θα πρέπει να συμπεριφέρεται σε διάφορες καταστάσεις, συχνά χρησιμοποιώντας τη μορφή 'Δεδομένου-Όταν-Τότε' ('Given-When-Then'). Αυτά τα σενάρια λειτουργούν τόσο ως προδιαγραφές όσο και ως περιπτώσεις δοκιμών, βοηθώντας να διασφαλιστεί ότι το αναπτυγμένο λογισμικό ανταποκρίνεται στις προβλεπόμενες απαιτήσεις και συμπεριφέρεται σωστά από την οπτική γωνία του χρήστη. [77, 59, 26]



```
1 describe("When creating a feature", () => {
2   describe("and the user is not authenticated", () => {
3     it("should throw an error", async () => {
4       ...

```

Σχήμα 2.22: Παράδειγμα μεθόδου που αφορά την δημιουργία ενός ελέγχου, με χρήση των μεθόδων *describe* και *it*

```
PASS  __tests__/api/game.test.ts
When creating a game
  and the user is not authenticated
    ✓ should throw an error (6 ms)
  and the user is authenticated
    and the user is not an admin
      ✓ should throw an error (1 ms)
    and the user is an admin
      and either the publisher or the franchise don't exist
        ✓ should return an error (4 ms)
    and the publisher and franchise exist
      ✓ should create a game successfully (3 ms)
```

Σχήμα 2.23: Παράδειγμα αποτελέσματος ελέγχου σε διεπαφή τερματικού, με χρήση των μεθόδων *describe* και *it*, μέσω του εργαλείου Jest

Επειδή το μοντέλο δουλεύει με next token prediction, και έλεγχοι γράφονται ο ένας μετά τον άλλο, το μοντέλο κατά τη συγγραφή των ελέγχων, λόγω των περιγραφών που περιέχονται στις μεθόδους, προσπαθούσε να γράψει τον έλεγχο με βάση την περιγραφή, αντί του κώδικα της μεθόδου, με αποτέλεσμα να οδηγείται συχνά σε λάθη και ανακρίβειες. Αξιοσημείωτη είναι η προσθήκη της δυνατότητας του GitHub Copilot Chat να γράψει ελέγχους για μια μέθοδο μέσω μιας εντολής στο πλαίσιο προτροπής με το μοντέλο [78] τον Μάρτιο του 2024, ωστόσο κατά την διάρκεια της δοκιμής του μοντέλου, η συγκεκριμένη δυνατότητα δεν ήταν διαθέσιμη.

Για την διαφορά μεταξύ αναγκαίων αλλαγών στις περιπτώσεις των αρχείων που αφορούσαν το Unit Testing και το Integration Testing, συνέβαλε και το γεγονός στην περίπτωση των ελέγχων μονάδας, οι έλεγχοι που πραγματοποιούνται είναι πιο απλοί και πολλές φορές χρησιμοποιούνται τεχνικές εικονικής αναπαράστασης εξαρτήσεων, γνωστές και ως τεχνικές προσομοίωσης εξαρτήσεων (dependency mocking). Η προσομοίωση εξαρτήσεων είναι μια τεχνική που χρησιμοποιείται στους ελέγχους λογισμικού, όπου δημιουργούνται ψεύτικα ή προσομοιωμένα αντικείμενα που μιμούνται τη συμπεριφορά πραγματικών εξαρτήσεων ενός συστήματος. Αυτό επιτρέπει στους προγραμματιστές να ελέγχουν μια μονάδα κώδικα σε απομόνωση, χωρίς να επηρεάζονται από εξωτερι-

κές εξαρτήσεις όπως βάσεις δεδομένων, υπηρεσίες δικτύου ή πολύπλοκες βιβλιοθήκες. Η προσομοίωση εξαρτήσεων βοηθά στη δημιουργία πιο ελεγχόμενων και προβλέψιμων συνθηκών δοκιμής, επιτρέποντας τον έλεγχο διαφόρων σεναρίων και καταστάσεων σφάλματος που μπορεί να είναι δύσκολο να αναπαραχθούν με πραγματικές εξαρτήσεις [28]. Αντίθετα, στην περίπτωση των ελέγχων ενσωμάτωσης, οι εξαρτήσεις είναι πραγματικές, και οι ελέγχοι που πραγματοποιούνται είναι πιο πολύπλοκοι και απαιτούν την ύπαρξη πραγματικών εξαρτήσεων, και χρειάζονται περισσότερη σκέψη για την υλοποίησή τους. Παρατίθενται δύο παραδείγματα ελέγχων, ένα από την περίπτωση των ελέγχων μονάδας και ένα από την περίπτωση των ελέγχων ενσωμάτωσης, για τον ίδιο έλεγχο.

```

1  describe("When creating a game", () => {
2    describe("and the user is authenticated", () => {
3      describe("and the user is an admin", () => {
4        describe("and the publisher and franchise exist", () => {
5          it("should create a game successfully", async () => {
6            // Arrange
7            const caller = appRouter.createCaller({
8              prisma: mockCtx.prisma,
9              session: mockAdminSession,
10            });
11
12            const gameId = createId();
13            const franchiseId = createId();
14
15            const gameData = {
16              name: faker.company.name(),
17              description: faker.lorem.words(),
18              coverImage: faker.image.url(),
19              backgroundImage: faker.image.url(),
20              releaseDate: new Date(),
21              franchiseId: franchiseId,
22              publisherId: gameId,
23            };
24
25            mockCtx.prisma.game.create.mockResolvedValue({
26              ...gameData,
27              id: gameId(),
28            });
29
30            // Act
31            const result = await caller.game.create(gameData);
32
33            // Assert
34            expect(result.ok).toBe(true);
35            expect(mockCtx.prisma.game.create).toHaveBeenCalledTimes(1);
36            expect(mockCtx.prisma.game.create).toHaveBeenCalledWith({
37              data: {
38                name: gameData.name,
39                description: gameData.description,
40                coverImage: gameData.coverImage,
41                backgroundImage: gameData.backgroundImage,
42                releaseDate: gameData.releaseDate,
43                franchise: {
44                  connect: {
45                    id: gameData.franchiseId,
46                  },
47                },
48                publisher: {
49                  connect: {
50                    id: gameData.publisherId,
51                  },
52                },
53              },
54            });
55          });
56        });
57      });
58    });
59  });

```

Σχήμα 2.24: Παράδειγμα ελέγχου μονάδας για την επιτυχημένη δημιουργία παιχνιδιού



```
1  describe("When creating a game", () => {
2      describe("and the user is an admin", () => {
3          describe("and the franchise and publisher exist", () => {
4              it("should create a game", async () => {
5                  // Arrange
6                  const franchise = await prisma.franchise.create({
7                      data: {
8                          name: faker.company.name(),
9                          // REWRITE_2: add description and image
10                         description: faker.company.catchPhrase(),
11                         image: faker.image.url(),
12                         },
13                     });
14
15                  const publisher = await prisma.publisher.create({
16                      data: {
17                          name: faker.company.name(),
18                          // REWRITE_2: add description and image
19                          description: faker.company.catchPhrase(),
20                          image: faker.image.url(),
21                          },
22                     });
23
24                  const game: z.infer<typeof createGameSchema> = {
25                      name: faker.company.name(),
26                      description: faker.company.catchPhrase(),
27                      coverImage: faker.image.url(),
28                      backgroundImage: faker.image.url(),
29                      releaseDate: new Date(),
30                      franchiseId: franchise.id,
31                      publisherId: publisher.id,
32                  };
33
34                  // Act
35                  const result = await adminCaller.game.create(game);
36
37                  // Assert
38                  expect(result.ok).toBe(true);
39                  expect(result.val).toMatchObject(game);
40              });
41          });
42      });
43  });
```

Σχήμα 2.25: Παράδειγμα ελέγχου συμαβότητας για την επιτυχημένη δημιουργία παιχνιδιού

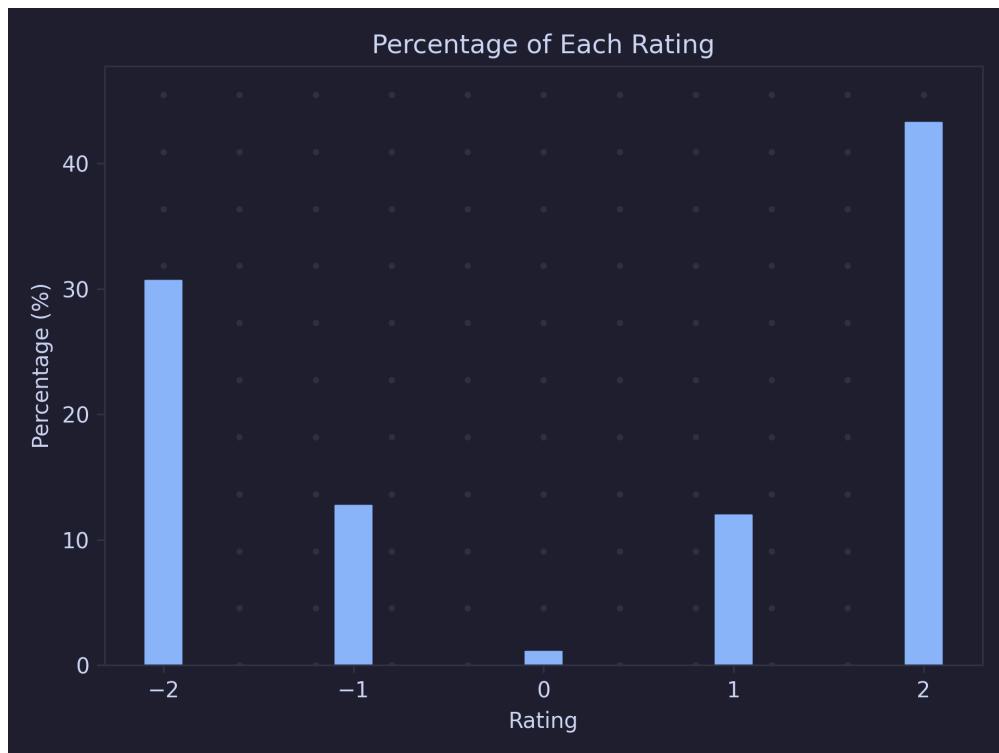
Εκ πρώτης όψεως, ο κώδικας στο Σχήμα ;; φαίνεται να είναι πιο περίπλοκος αυτού στο Σχήμα ;;, ωστόσο, στην περίπτωση του ελέγχου μονάδας, οι εξαρτήσεις είναι προσομοιωμένες. Δεν υπάρχει κάποιος πραγματικός χρήστης, ή κάποια πραγματική βάση δεδομένων. Αντίθετα, στην περίπτωση του ελέγχου συμβατότητας, οι εξαρτήσεις είναι πραγματικές, και ο έλεγχος πρέπει να ελέγξει την συμβατότητα του συστήματος με μια πραγματική βάση δεδομένων, ένα πραγματικό χρήστη και μετά από τον κάθε έλεγχο, πρέπει να είναι σίγουρο ότι εσωτερική κατάσταση του προγράμματος και της βάσης είναι κοινή για όλες τις περιπτώσεις, αποφεύγοντας περιπτώσεις ασταθών δοκιμών (flaky tests), όπου το αποτέλεσμα της δοκιμής δεν είναι πάντοτε αμετάβλητο. [67]

2.4 Αξιολόγηση Απαντήσεων

Κατά την αλληλεπίδραση με το μοντέλο, για κάθε απάντηση που δόθηκε έπειτα προτροπής, πραγματοποιήθηκε μια αξιολόγηση σε μια κλίμακα μεταξύ μείον δύο (-2) και δύο (2), με το μείον δύο να αντιστοιχεί σε μια απάντηση η οποία κρίθηκε ως εντελώς λάθος, το μείον ένα (-1) σε μια απάντηση η οποία κρίθηκε ως μερικώς λάθος, το μηδέν (0) σε μια απάντηση η οποία κρίθηκε ως αδιάφορη προς την ορθότητά της, το ένα (1) σε μια απάντηση η οποία κρίθηκε ως σωστή και το δύο (2) σε μια απάντηση η οποία κρίθηκε ως εντελώς σωστή.

Η επιλογή μιας κλίμακας η οποία εκτείνεται από ‘πολύ κακό’ έως ‘πολύ καλό’, αποσκοπεί στην καλύτερη αναπαράσταση της ποικιλίας και της πολυπλοκότητας της αξιολόγησης μιας απάντησης και στην δημιουργία ενός συνόλου ασαφούς λογικής (fuzzy logic) για την αξιολόγηση των απαντήσεων [96, 48, 74].

Η αξιολόγηση των απαντήσεων αποσκοπεί στην εξάσκηση ενός μοντέλου μηχανικής μάθησης, προκειμένου να βελτιωθεί η απόδοσή του στην παραγωγή απαντήσεων, με βάση την ορθότερη προτροπή, του οποίου η ανάλυση γίνεται στο επόμενο κεφάλαιο.



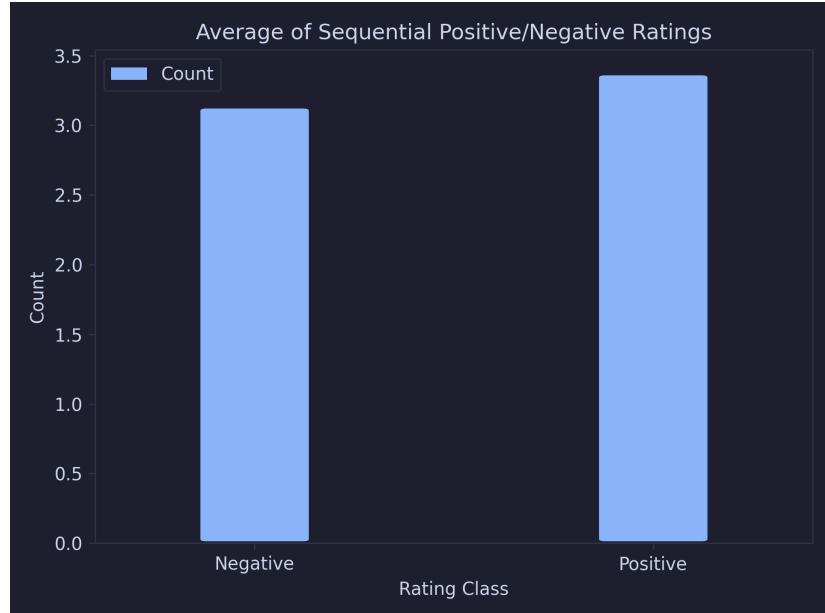
Σχήμα 2.26: Ποσοστό αξιολόγησης απαντήσεων



Σχήμα 2.27: Αριθμός αξιολόγησης απαντήσεων

Από τα σχήματα ;; και ;;, παρατηρείται ότι η πλειοψηφία των απαντήσεων αξιολογήθηκαν θετικά, με το το πενήντα πέντε τοις εκατό (55.3435%), με σύνολο 290 θετικών

αξιολογήσεων, έναντι του σαράντα τρία τοις εκατό (43.5115%), με σύνολο 228 αρνητικών αξιολογήσεων. Ο συνηθέστερος βαθμός αξιολόγησης ήταν το δύο (2), με σύνολο διακοσίων εικοσιεπτά (227) αξιολογήσεων, ενώ ο σπανιότερος βαθμός αξιολόγησης ήταν το μηδέν (0), με σύνολο έξι (6) αξιολογήσεων.



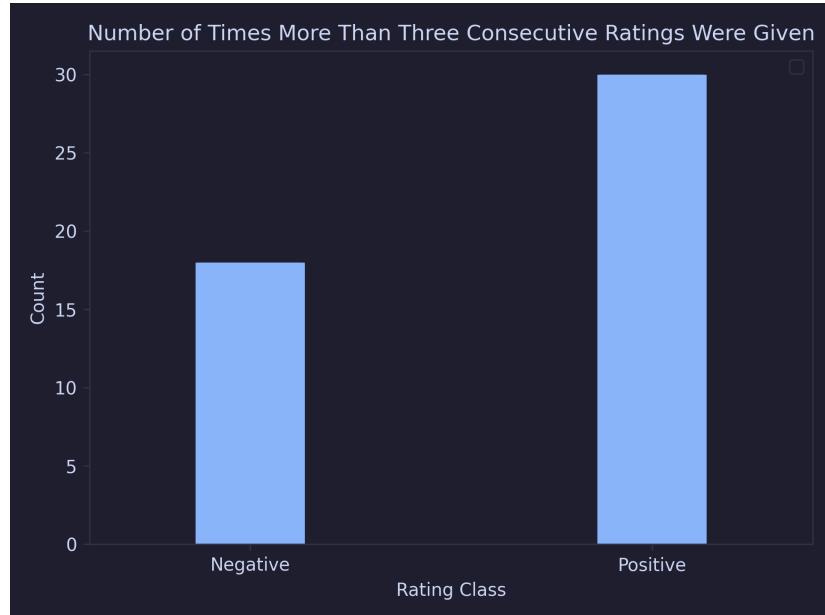
Σχήμα 2.28: Μέσος όρος συνεχόμενης αξιολόγησης απαντήσεων



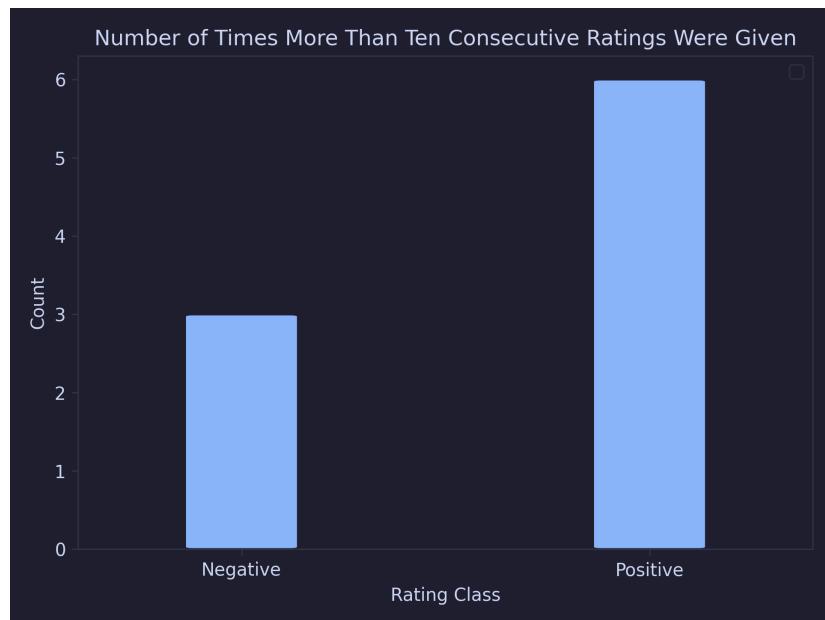
Σχήμα 2.29: Μέγιστη συνεχόμενη αξιολόγηση απαντήσεων

Το μοντέλο φαίνεται να έχει την τάση να παράγει απαντήσεις οι οποίες βασίζονται πολύ στην αρχική παρότρυνση από την προτροπή του προγραμματιστή. Αυτό αποδεικνύεται και από το σχήμα ;, όπου παρατηρείται ότι ο μέγιστος αριθμός συνεχόμενων αρνητικών αξιολογήσεων είναι δεκατέσσερεις (14), ενώ ο μέγιστος αριθμός συνεχόμενων θετικών

αξιολογήσεων είναι δεκατρείς (16). Στην περίπτωση των συνεχόμενων θετικών αξιολογήσεων, οι προτροπές αφορούσαν 3 διαφορετικά θέματα, τους ελέγχους της λειτουργίας των κριτικών, την ανάπτυξη της λειτουργίας των λιστών, καθώς και τους ελέγχους της. Στην περίπτωση των συνεχόμενων αρνητικών αξιολογήσεων, οι προτροπές αφορούσαν μόνο τους ελέγχους μιας λειτουργίας.



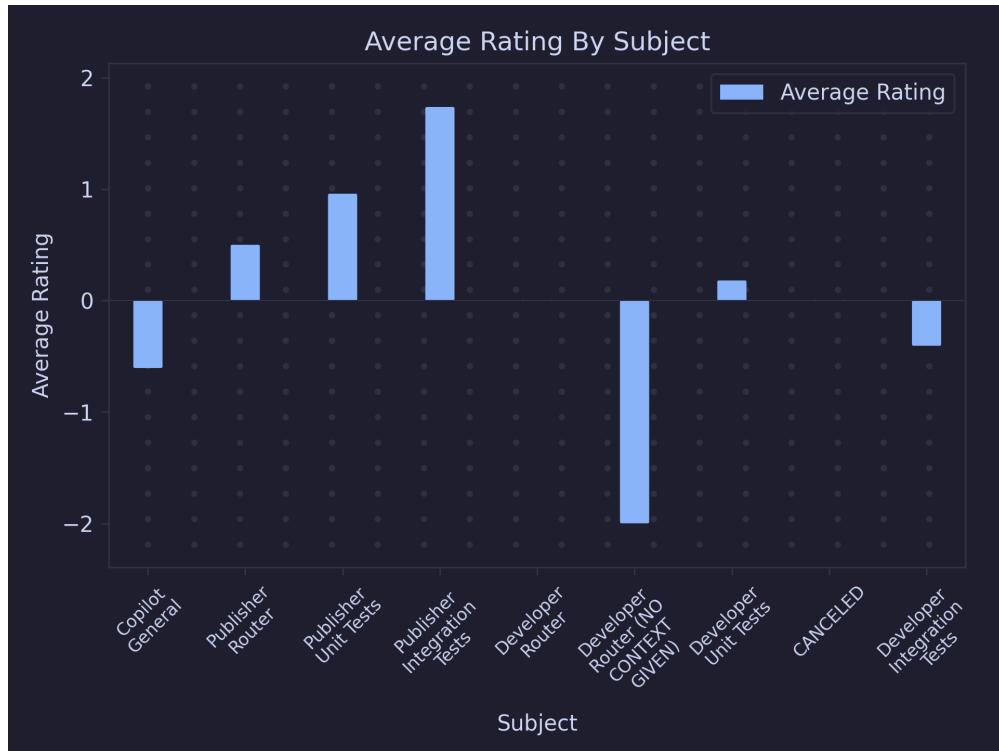
Σχήμα 2.30: Αριθμός εμφανίσεων συνεχόμενων αξιολογήσεων άνω των τριών (3)



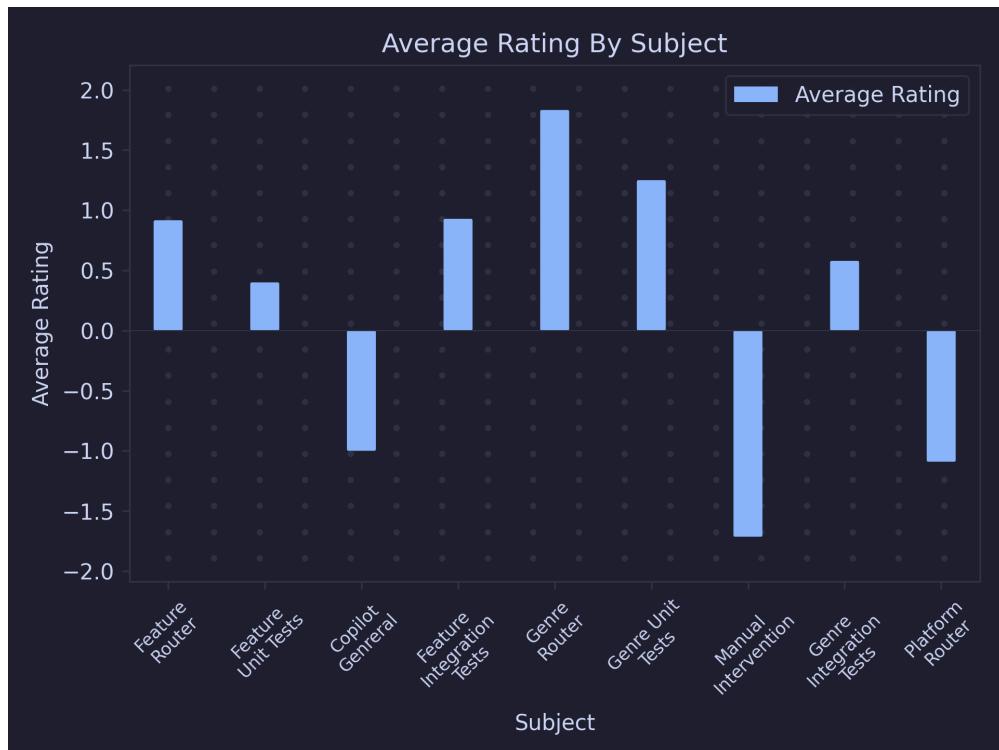
Σχήμα 2.31: Αριθμός εμφανίσεων συνεχόμενων αξιολογήσεων άνω των δέκα (10)

Για την περίπτωση των άνω των δέκα (10) συνεχόμενων θετικών αξιολογήσεων, κατά μέσο όρο, αφορούσαν δύο (2) διαφορετικά θέματα, ενώ ήταν διπλάσιες οι περιπτώσεις εμφάνισης των συνεχόμενων θετικών έναντι των αρνητικών αξιολογήσεων (;;), που αφορούσαν ένα θέμα. Φαίνεται πως αν η πρώτη προτροπή από τον προγραμματιστή δεν

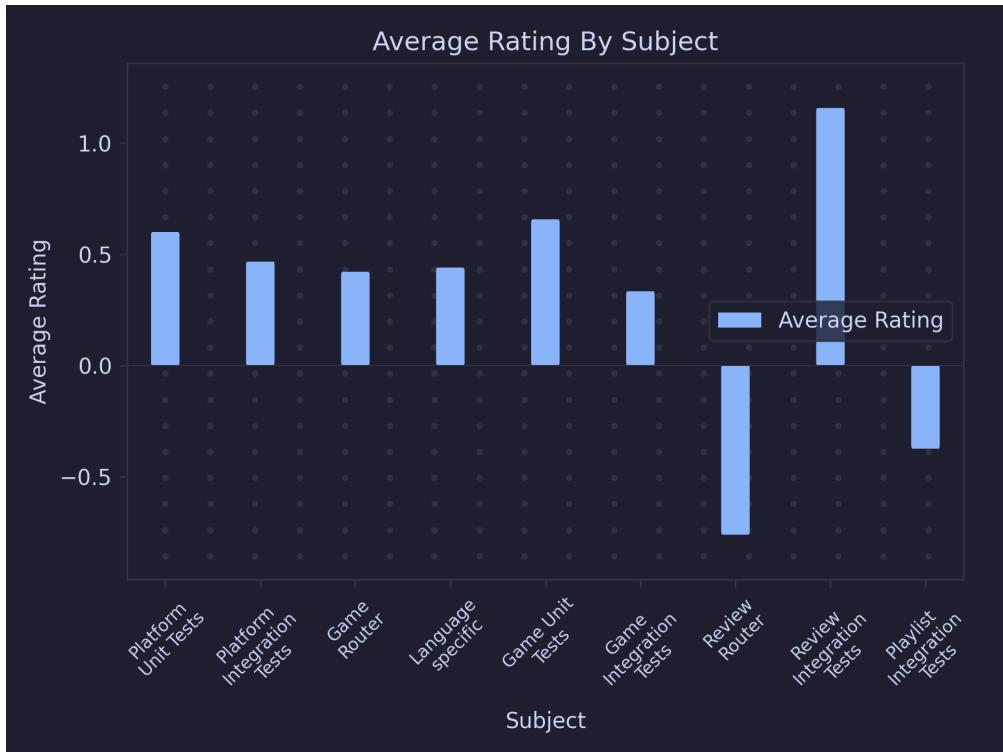
οδηγήσει ορθά το μοντέλο, τότε δύσκολα βρίσκει τη λύση με τις επόμενες προτροπές, καθώς διατηρείται η αρχική κατεύθυνση που έλαβε.



Σχήμα 2.32: Μέσος όρος αξιολόγησης απαντήσεων για θέματα, μέρος 1



Σχήμα 2.33: Μέσος όρος αξιολόγησης απαντήσεων για θέματα, μέρος 2



Σχήμα 2.34: Μέσος όρος αξιολόγησης απαντήσεων για θέματα, μέρος 3

Από τα σχήματα ::, :: και ::, παρατηρείται ότι δεν υπάρχει κάποια συγκεκριμένη τάση στην αξιολόγηση των απαντήσεων ανά θέμα, με τα περίσσοτερα των θεμάτων να έχουν θετικό μέσο όρο. Αξιοσημείωτη η περίπτωση του θέματος της ανάπτυξης της λειτουργίας του προγραμματιστή του παιχνιδιού (Developer Router), όπου στις πρώτες προτροπές, δεν δόθηκε στο μοντέλο κάποια κατεύθυνση, με αποτέλεσμα οι απαντήσεις που παρήχθηκαν να αξιολογήθουν αρνητικά, ενώ στις επόμενες προτροπές, οι απαντήσεις που παρήχθηκαν αξιολογήθηκαν κατά μέσο όρο ίσο με μηδέν (0), παρατηρώντας την μεγαλύτερη αύξηση στο μέσο όρο αξιολογήσεων από θέμα σε θέμα.

Το θέμα Manual Intervention αφορούσε τις περιπτώσεις που ο προγραμματιστής αναγκάστηκε να επενέβει, με σκοπό να αποφευχθούν συνεχόμενες αρνητικές αξιολογήσεις στο ίδιο θέμα μετά από μια σειρά λάθος απαντήσεων.

2.5 Συμπεράσματα

Όντας μια καινοτόμος επιλογή τεχνολογικής στοίβας, η χρήση της Typescript, σε συνδυασμό με το tRPC και το Prisma, αποτέλεσε μια πρόκληση για το μοντέλο του GitHub Copilot. Η επιλογή αυτή έγινε με σκοπό την ορθότερη αξιολόγηση της ικανότητας του μοντέλου να ακολουθήσει την ταχεία εξέλιξη των τεχνολογιών και συγκεκριμένα στο οικοσύστημα της Javascript, καθώς έχει συνεχείς προσθήκες και αλλαγές στον τρόπο που γράφεται ο κώδικας και επίσης αποτελεί τη βασική προγραμματιστική γλώσσα του διαδικτύου.

Εκ του αποτελέσματος των δοκιμών, η ιδανικότερη χρήση του μοντέλου του GitHub Copilot φαίνεται είναι να ακολουθήσει τις επιλογές του προγραμματιστή για την ανάπτυξη του κώδικα, την αρχιτεκτονική, τα πρότυπα σχεδίασης και τις βιβλιοθήκες που

Θα χρησιμοποιηθούν κατά την ανάπτυξη της εφαρμογής, όπως και αποδείχθηκε από τα αποτελέσματα στις περιπτώσεις που δόθηκε η ευκαιρία στο μοντέλο να λάβει αποφάσεις, καθώς οι περισσότερες αλλαγές που πραγματοποιήθηκαν από την πλευρά του προγραμματιστή ήταν σε μεγάλες εκτάσεις κώδικα και συγκεκριμένα στις περιπτώσεις των ελέγχων, όπου εκεί το μοντέλο έλαβε πρωτοβουλίες κατά τη σύνταξη των ελέγχων, οδηγώντας σε λάθη.

Κεφάλαιο 3

Μοντέλο πρόβλεψης απόδοσης απαντήσεων βάσει προτροπής

Στο Κεφάλαιο 2 παρουσιάστηκε η μεθοδολογία που ακολουθήθηκε για την συλλογή των δεδομένων και την αξιολόγηση των απαντήσεων του μοντέλου του GitHub Copilot. Η κατηγοριοποίηση των δεδομένων, οδήγησε στην ιδέα να αναπτυχθεί ένα μοντέλο μηχανικής μάθησης με σκοπό την πρόβλεψη της απόδοσης μιας απάντησης του μοντέλου του GitHub Copilot, με βάση την προτροπή.

Με βάση την προτροπή, αρχικά έγινε η προσπάθεια για την πρόβλεψη της ακριβούς τιμής της αξιολόγησης, από πολύ αρνητική έως πολύ θετική, αλλά, λόγω του μικρού μεγέθους των δεδομένων, η παραπάνω λογική οδήγησε σε εξαιρετικά ασυνεπή αποτέλεσματα, με ακρίβεια χαμηλότερη αυτής των πενήντα τοις εκατό (50%). Συνεπώς, αποφασίστηκε να γίνει η προσπάθεια κατηγοριοποίησης της απάντησης είτε σε θετική αξιολόγηση, είτε σε αρνητική.

3.1 Κατηγοριοποίηση των δεδομένων

Για την κατηγοριοποίηση των δεδομένων, η κάθε προτροπή επεξεργάστηκε, προκειμένου να αφαιρεθούν τα κομμάτια κώδικα από την υπόλοιπη προτροπή και έπειτα αφαιρέθηκαν οι "κενές λέξεις" (stopwords) με την χρήση της βιβλιοθήκης Natural Language Toolkit (NLTK). Για κάθε προτροπή που είχε κομμάτια κώδικα, ο αριθμός των συμβόλων (symbols), αποθηκεύθηκε σε μια άλλη κατηγορία.

Ένα από τα μεγαλύτερα προβλήματα των Γλωσσικών Μοντέλων, είναι ο μέγιστος αριθμός των συμβόλων (tokens) που μπορούν να επεξεργαστούν. Καθώς οι προτροπές ζητούσαν μεγάλα κώδικα, ειδικά στις περιπτώσεις των ελέγχων του κώδικα, το μοντέλο συχνά έφτανε στο όριο των δεδομένων που μπορούσε να παράξει στην απάντησή του. Καθώς το μοντέλο δεν γνώριζε εκ των προτέρων το μέγεθος της απάντησης, αν ο προγραμματιστής δεν το παρότρυνε να σταματήσει σε ένα συγκεκριμένο αριθμό συμβόλων, το μοντέλο θα συνέχιζε να παράγει κώδικα μέχρι να φτάσει στο όριο των δεδομένων που μπορούσε να παράξει, οδηγώντας σε μια σειρά από αποτυχημένες απαντήσεις.

Για την αντιμετώπιση του προβλήματος, ο προγραμματιστής ζήτησε από το μοντέλο να απαντάει για ένα συγκεκριμένο μέρος του προβλήματος, όπως παραδείγματος χάρη έναν συγκεκριμένο έλεγχο, αντί όλων των ελέγχων. Έπειτα, ζήτησε από το μοντέλο να

περιμένει την προτροπή με περιεχόμενο την λέξη "continue" (συνέχισε), προκειμένου να συνεχίσει με την προηγούμενη απάντησή του. Με αυτόν τον τρόπο, το μοντέλο θα μπορούσε να απαντήσει σε μια προτροπή, χωρίς να φτάσει στο όριο των δεδομένων που μπορούσε να παράξει.

Ωστόσο, αυτό οδήγησε σε αρκετές προτροπές με μοναδικό περιεχόμενο την λέξη "continue", που αφενός δεν είχαν καμία σχέση με το προηγούμενο κείμενο και αφετέρου, εφ' όσον ο προγραμματιστής παρότρυνε το μοντέλο να συνεχίσει με την απάντησή του, σήμαινε πως η τουλάχιστον αμέσως προηγούμενη απάντηση ήταν ικανοποιητική, οδηγώντας σε μια μεροληψία προς θετικά αποτελέσματα. Συνεπώς, αποφασίστηκε να υπολογίζεται ο συνεχόμενος αριθμός προτροπών με περιεχόμενο την λέξη "continue", συνάμα με το θέμα της προτροπής, ώστε να δημιουργούνται ομάδες προτροπών που στοχεύουν στο ίδιο ερώτημα, αλλά χρειάστηκε να διασπαστούν λόγω των περιορισμών του μοντέλου.

Περιπτώσεις προτροπών που είτε ματαιώθηκαν, είτε δεν είχαν απάντηση λόγω τεχνικών προβλημάτων, ήταν σημειωμένες εξαρχής και παραλείφθηκαν από την ανάλυση.

3.1.1 Προεπεξεργασία και Προετοιμασία Δεδομένων

Με την χρήση των βιβλιοθηκών πανδας και σικιτ-λεαρν, η διαδικασία προεπεξεργασίας και προετοιμασίας των δεδομένων περιελάμβανε τα ακόλουθα στάδια:

- Εισαγωγή Δεδομένων:** Τα δεδομένα εισήχθησαν από ένα αρχείο "Σ" χρησιμοποιώντας τη βιβλιοθήκη πανδας για τον αποτελεσματικό χειρισμό τους.
- Κωδικοποίηση Κατηγορικών Μεταβλητών:** Οι κατηγορικές μεταβλητές "subject" (θέμα) και "type" (τύπος) κωδικοποιήθηκαν αριθμητικά με τη χρήση του LabelEncoder.
- Μετατροπή Χρονικών Δεδομένων:** Η στήλη "createdAt" μετατράπηκε σε αριθμητική μορφή, αντιπροσωπεύοντας τα δευτερόλεπτα από την εποχή UNIX (1/1/1970).
- Δυαδικοποίηση Αξιολογήσεων:** Η στήλη "rating" (αξιολόγηση) μετατράπηκε σε δυαδική μορφή, όπου οι θετικές αξιολογήσεις αντιστοιχήθηκαν στην τιμή ένα (1) και οι αρνητικές στην τιμή μηδέν (0).
- Ομαδοποίηση Δεδομένων:** Πραγματοποιήθηκε ομαδοποίηση των δεδομένων με βάση τη στήλη "group". Για κάθε ομάδα, υπολογίστηκαν διάφορα στατιστικά μέτρα, όπως ο μέσος όρος για τις αριθμητικές μεταβλητές και η επικρατούσα τιμή για τις κατηγορικές.
- Επεξεργασία Κειμένου:** Εφαρμόστηκε η τεχνική Term Frequency-Inverse Document Frequency (TF-IDF) για τη μετατροπή των κειμενικών προτροπών σε αριθμητικά χαρακτηριστικά. Η στήλη "prompt" (προτροπή) μετασχηματίστηκε σε πολλαπλές στήλες, κάθε μία εκ των οποίων αντιπροσωπεύει ένα διακριτό χαρακτηριστικό TF-IDF.
- Διαχείριση Ελλιπών Τιμών:** Οι ελλιπείς τιμές στις αριθμητικές στήλες αντικαταστάθηκαν με τη μέση τιμή της αντίστοιχης στήλης, διασφαλίζοντας την πληρότητα του συνόλου δεδομένων.

3.1.2 Επιλογή Χαρακτηριστικών

Η διαδικασία επιλογής χαρακτηριστικών περιελάμβανε τα εξής βήματα:

1. **Εφαρμογή SelectKBest:** Χρησιμοποιήθηκε η μέθοδος SelectKBest σε συνδυασμό με τη συνάρτηση (χ^2) για την επιλογή των δέκα (10) πιο σημαντικών χαρακτηριστικών. Αυτή η προσέγγιση συνέβαλε στη μείωση της διαστατικότητας του προβλήματος και στην επικέντρωση στα πλέον πληροφοριακά χαρακτηριστικά.
2. **Διαχωρισμός Δεδομένων:** Τα δεδομένα διαχωρίστηκαν σε σύνολα εκπαίδευσης και ελέγχου, με αναλογία ογδόντα τοις εκατό (80%) και είκοσι τοις εκατό (20%) αντίστοιχα, χρησιμοποιώντας τη συνάρτηση train_test_split.

3.2 Εκπαίδευση και Αξιολόγηση Μοντέλων

Για την εκπαίδευση και αξιολόγηση των μοντέλων ακολουθήθηκε η εξής διαδικασία:

1. **Επιλογή Αλγορίθμων:** Επιλέχθηκαν τέσσερις διαφορετικοί αλγόριθμοι μηχανικής μάθησης: Random Forest, Support Vector Machine (SVM), Gradient Boosting, και Naive Bayes.
2. **Βελτιστοποίηση Υπερπαραμέτρων:** Για κάθε αλγόριθμο, πραγματοποιήθηκε εκτενής αναζήτηση πλέγματος (Grid Search) με δεκάπτυχη διασταυρωμένη επικύρωση για τη βελτιστοποίηση των υπερπαραμέτρων.
3. **Εκπαίδευση Μοντέλων:** Τα μοντέλα εκπαιδεύτηκαν χρησιμοποιώντας τις βέλτιστες υπερπαραμέτρους που προέκυψαν από την αναζήτηση πλέγματος.
4. **Αξιολόγηση Μοντέλων:** Η απόδοση των μοντέλων αξιολογήθηκε χρησιμοποιώντας το σύνολο ελέγχου. Υπολογίστηκαν μετρικές όπως η ακρίβεια, η ανάκληση, και το F1-score.
5. **Οπτικοποίηση Αποτελεσμάτων:** Δημιουργήθηκαν πίνακες σύγχυσης (confusion matrices) και συγκριτικά γραφήματα για την οπτική αναπαράσταση της απόδοσης των διαφόρων μοντέλων.

Η μεθοδολογία αυτή επέτρεψε τη συστηματική ανάπτυξη, εκπαίδευση και αξιολόγηση των μοντέλων, παρέχοντας μια ολοκληρωμένη εικόνα της απόδοσής τους στην ταξινόμηση των δεδομένων.

Πραγματοποιήθηκε επίσης έρευνα για την επιλογή ανάπτυξης ενός νευρωνικού δικτύου, αλλά λόγω του μικρού μεγέθους των δεδομένων, το αποτέλεσμα ήταν ανεπαρκή, με ακρίβεια κάτω του πενήντα τοις εκατό (50%). Με την εφαρμογή των παραπάνω βημάτων, η ακρίβεια των μοντέλων αυξήθηκε σημαντικά, με τα αποτελέσματα να αναλύονται παρακάτω.

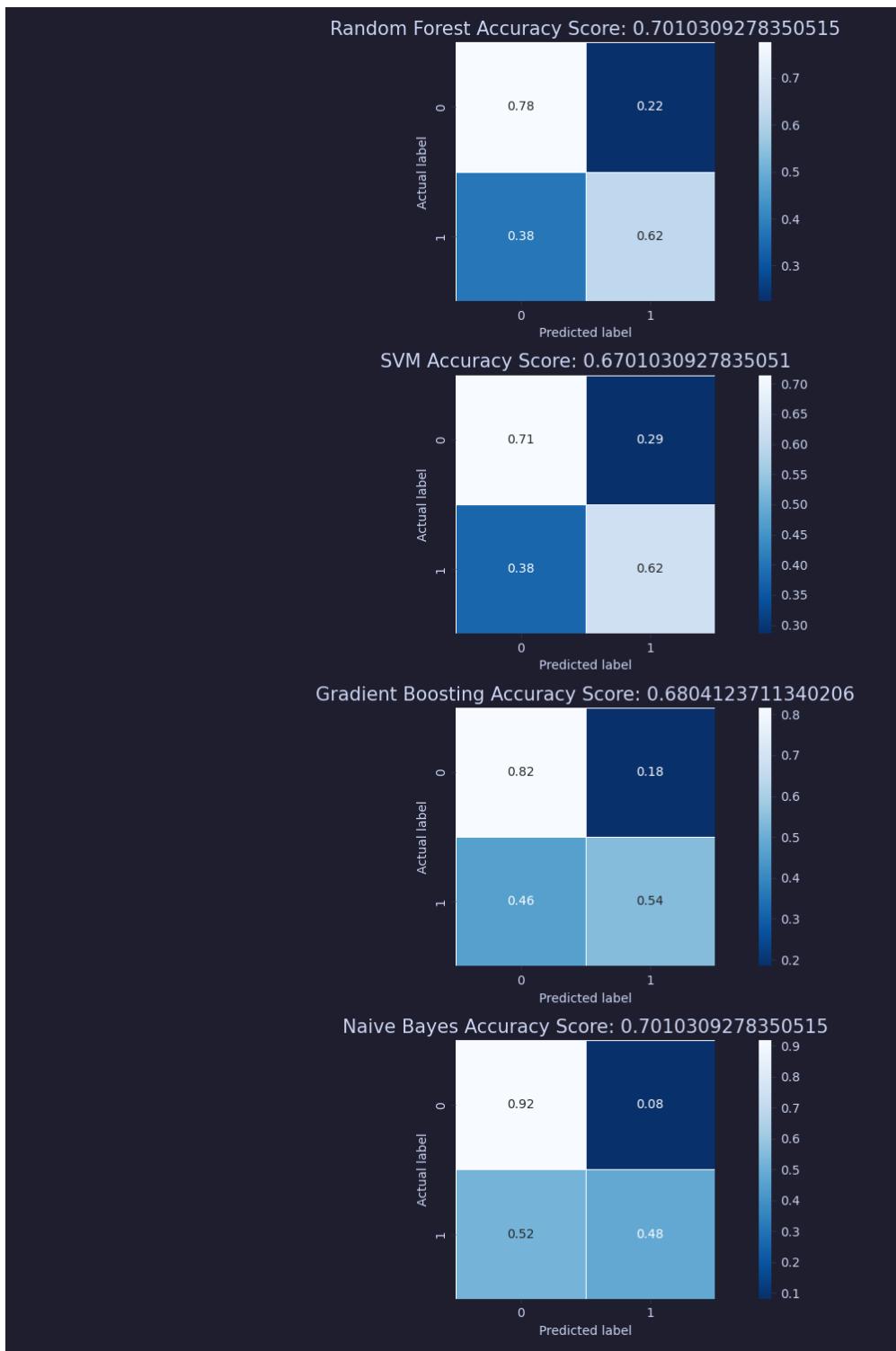
3.3 Αποτελέσματα

Η ανάπτυξη ενός μοντέλου πρόβλεψης της απόδοσης Γλωσσικών Μοντέλων βάσει πολλαπλών κριτηρίων, με ιδιαίτερη έμφαση στην προτροπή, αποτελεί ένα καινοτόμο ερευνητικό εγχείρημα. Ο όρος "μηχανική προτροπής" (Prompt Engineering) αναφέρεται

στη εξειδικευμένη μεθοδολογία σχεδιασμού και βελτιστοποίησης προτροπών με στόχο τη μεγιστοποίηση της απόδοσης του μοντέλου [43, 20].

Οι επιστημονικές έρευνες στον ραγδαία εξελισσόμενο τομέα αυτό είναι ιδιαίτερα πρόσφατες, με την πλειονότητα των σημαντικών δημοσιεύσεων να εμφανίζεται μετά τη δημόσια κυκλοφορία του ChatGPT [36;]. Ωστόσο, η αποτελεσματική εφαρμογή και βελτιστοποίηση της μηχανικής προτροπής αποτελεί ένα από τα πιο κρίσιμα και πολύπλοκα προβλήματα ενός κόσμου ολοένα και περισσότερο ενισχυμένου από την τεχνητή νοημοσύνη.

Η εξαγωγή του μέγιστου δυνατού επιπέδου απόδοσης από τα Γλωσσικά Μοντέλα, και κατ' επέκταση η μείωση σφαλμάτων και μη ικανοποιητικών αποκρίσεων, με αποτέλεσμα την αποτελεσματική αύξηση της απόδοσης χωρίς την εισαγωγή επιπρόσθετων δεδομένων ή την τροποποίηση του υπάρχοντος κάδικα ανάπτυξης του μοντέλου, αποτελεί μια διαδικασία τόσο δύσκολη όσο και δυσνόητη. Η αξιοποίηση ενός μοντέλου μηχανικής μάθησης για την πρόβλεψη της απόδοσης ενός γλωσσικού μοντέλου βάσει της προτροπής δύναται να παράσχει πολύτιμα εργαλεία για τη μεγιστοποίηση της απόδοσης.



Σχήμα 3.1: Μέσος όρος αξιολόγησης απαντήσεων για θέματα, μέρος 3

Παράτημα Α'

Ακρωνύμια και συντομογραφίες

LLM Large Language Model

NLP Natural Language Processing

IDE Integrated Development Environment

LSP Language Server Protocol

API Application Programming Interface

ORM Object Relational Mapping

CRUD Create Read Update Delete

BDD Behavior Driven Development

NLTK Natural Language Toolkit

TF-IDF Frequency-Inverse Document Frequency

SVM Support Vector Machine

Bibliography

- [1] “Faker.js.” [Online]. Available: <https://fakerjs.dev/>
- [2] “Github copilot.” [Online]. Available: <https://github.com/features/copilot>
- [3] “Github student developer pack.” [Online]. Available: <https://education.github.com/pack>
- [4] “Lorem ipsum - all the facts - lipsum generator,” <https://www.lipsum.com/>, retrieved June 11, 2024.
- [5] *Guide to the Software Engineering Body of Knowledge*, 2004, a project of the IEEE Computer Society Professional Practices Committee.
- [6] “Trendforce says with cloud companies initiating ai arms race, gpu demand from chatgpt could reach 30,000 chips as it readies for commercialization,” *TrendForce*, Nov 2023, archived from the original on May 30, 2024. Retrieved November 2, 2023.
- [7] “Visual studio code.” [Online]. Available: <https://code.visualstudio.com/>
- [8] S. Adams, I. Arel, J. Bach, R. Coop, R. Furlan, B. Goertzel, J. S. Hall, A. Samsonovich, M. Scheutz, M. Schlesinger *et al.*, “Mapping the landscape of human-level artificial general intelligence,” *AI Magazine*, vol. 33, no. 1, pp. 25–42, 2012.
- [9] A. Ait Baha, M. El Hajji, Y. Es-saady, and H. Fadili, “The power of personalization: A systematic review of personality-adaptive chatbots,” *SN Computer Science*, vol. 4, 2023. [Online]. Available: <https://api.semanticscholar.org/CorpusID:261346287>
- [10] “Javascriptcore,” Apple. [Online]. Available: <https://docs.webkit.org/Deep%20Dive/JSC/JavaScriptCore.html>
- [11] “Webkit,” Apple. [Online]. Available: <https://webkit.org/>
- [12] M. Asaduzzaman, C. K. Roy, K. A. Schneider, and D. Hou, “Csc: Simple, efficient, context sensitive code completion,” in *2014 IEEE International Conference on Software Maintenance and Evolution*. IEEE, 2014, pp. 71–80.
- [13] Atlassian, “Join relationships and joining tables,” <https://www.atlassian.com/data/sql/joins>, retrieved June 11, 2024.
- [14] T. Ball, *Writing An Interpreter In Go*. Thorsten Ball, 2018.

- [15] J. Bays, “Aws announces amazon codewhisperer (preview),” Jun 2022. [Online]. Available: <https://aws.amazon.com/about-aws/whats-new/2022/06/aws-announces-amazon-codewhisperer-preview/>
- [16] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill *et al.*, “On the opportunities and risks of foundation models,” 2021.
- [17] H. Bunder, “Decoupling language and editor - the impact of the language server protocol on textual domain-specific languages,” in *Proceedings of the 7th International Conference on Model-Driven Engineering and Software Development - MODELSWARD*, INSTICC. SciTePress, 2019, pp. 131–142.
- [18] T. S. BV, “Tiobe index for javascript,” <https://www.tiobe.com/tiobe-index/javascript/>, retrieved June 01 2024.
- [19] S. Carter, “Exponential baby! navigating the ai convergence of tech with nvidia,” *Forbes Digital Assets*, May 2024, retrieved May 30, 2024. [Online]. Available: <https://www.forbes.com/sites/digital-assets/2024/05/11/exponential-baby-navigating-the-ai--convergence-of-tech-with-nvidia/>
- [20] B. Chen, Z. Zhang, N. Langrené, and S. Zhu, “Unleashing the potential of prompt engineering in large language models: a comprehensive review,” 2024. [Online]. Available: <https://arxiv.org/abs/2310.14735>
- [21] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. de Oliveira Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, A. Ray, R. Puri, G. Krueger, M. Petrov, H. Khlaaf, G. Sastry, P. Mishkin, B. Chan, S. Gray, N. Ryder, M. Pavlov, A. Power, L. Kaiser, M. Bavarian, C. Winter, P. Tillet, F. P. Such, D. Cummings, M. Plappert, F. Chantzis, E. Barnes, A. Herbert-Voss, W. H. Guss, A. Nichol, A. Paino, N. Tezak, J. Tang, I. Babuschkin, S. Balaji, S. Jain, W. Saunders, C. Hesse, A. N. Carr, J. Leike, J. Achiam, V. Misra, E. Morikawa, A. Radford, M. Knight, M. Brundage, M. Murati, K. Mayer, P. Welinder, B. McGrew, D. Amodei, S. McCandlish, I. Sutskever, and W. Zaremba, “Evaluating large language models trained on code,” 2021.
- [22] M. Ciniselli, N. Cooper, L. Pascarella, A. Mastropaoletti, E. Aghajani, D. Poshyvanyk, M. Di Penta, and G. Bavota, “An empirical study on the usage of transformer models for code completion,” *IEEE Transactions on Software Engineering*, 2021.
- [23] M. Ciniselli, N. Cooper, L. Pascarella, D. Poshyvanyk, M. Di Penta, and G. Bavota, “An empirical study on the usage of bert models for code completion,” in *2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)*, 2021, pp. 108–119.
- [24] D. Crockford, *JavaScript: The Good Parts*, 1st ed., ser. Yahoo Press. Sebastopol, CA: O'Reilly Media, 2008, pp. 36–40.
- [25] T. Dohmke, “Github copilot x: The ai-powered developer experience,” *GitHub Blog*, March 2023, retrieved May 30, 2024. [Online]. Available: <https://github.blog/2023-03-22-github-copilot-x-the-ai-powered-developer-experience/>

- [26] S. Farooq, U. Omer, A. Ramzan, M. Rasheed, and Z. Atal, “Behavior driven development: A systematic literature review,” *IEEE Access*, vol. PP, pp. 1–1, 01 2023.
- [27] Z. Feng, D. Guo, D. Tang, N. Duan, X. Feng, M. Gong, L. Shou, B. Qin, T. Liu, D. Jiang, and M. Zhou, “Codebert: A pre-trained model for programming and natural languages,” in *Findings of the Association for Computational Linguistics: EMNLP 2020*. Online: Association for Computational Linguistics, Nov 2020, pp. 1536–1547.
- [28] S. Freeman and N. Pryce, *Growing Object-Oriented Software, Guided by Tests*. Upper Saddle River, NJ: Addison-Wesley Professional, 2009.
- [29] D. Fried, A. Aghajanyan, J. Lin, S. Wang, E. Wallace, F. Shi, R. Zhong, S. Yih, L. Zettlemoyer, and M. Lewis, “Incoder: A generative model for code infilling and synthesis,” in *The Eleventh International Conference on Learning Representations*, 2023.
- [30] A. Gabillon, M. Munier, J. J. Bascou, L. Gallon, and E. Bruno, “An access control model for tree data structures,” in *Information Security, 5th International Conference, ISC 2002 Sao Paulo, Brazil, September 30 - October 2, 2002, Proceedings*, Sao Paulo, Brazil, 2002, pp. 117–135. [Online]. Available: <https://hal.archives-ouvertes.fr/hal-01912344>
- [31] “Github copilot chat in visual studio code,” <https://github.com/microsoft/vscode-copilot-release>, GitHub, 2023.
- [32] GitHub, “Introducing github copilot: your ai pair programmer,” *GitHub Blog*, June 2021, retrieved May 30, 2024. [Online]. Available: <https://github.blog/2021-06-29-introducing-github-copilot-ai-get-code-done-faster/>
- [33] B. Goertzel, “Artificial general intelligence: Concept, state of the art, and future prospects,” *Journal of Artificial General Intelligence*, vol. 5, no. 1, pp. 1–46, 2014, submitted 2013-2-12, Accepted 2014-3-15.
- [34] “Chromium,” Google. [Online]. Available: <https://www.chromium.org/Home>
- [35] “V8 javascript engine,” Google. [Online]. Available: <https://v8.dev>/
- [36] K. Greshake, S. Abdehnabi, S. Mishra, C. Endres, T. Holz, and M. Fritz, “Not what you’ve signed up for: Compromising real-world llm-integrated applications with indirect prompt injection,” 2023. [Online]. Available: <https://arxiv.org/abs/2302.12173>
- [37] D. Guo, S. Lu, N. Duan, Y. Wang, M. Zhou, and J. Yin, “Unircoder: Unified cross-modal pre-training for code representation,” in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 7212–7225.
- [38] J. Irena, *Software Testing Methods and Techniques*, 2008.
- [39] M. Izadi, R. Gismondi, and G. Gousios, “Codefill: Multi-token code completion by jointly learning from structure and naming sequences,” in *Proceedings of*

the 44th International Conference on Software Engineering, ser. ICSE '22. New York, NY, USA: Association for Computing Machinery, 2022, pp. 401–412.

- [40] M. Izadi, J. Katzy, T. van Dam, M. Otten, R. M. Popescu, and A. van Deursen, “Language models for code completion: A practical evaluation,” in *2024 IEEE/ACM 46th International Conference on Software Engineering (ICSE '24)*. New York, NY, USA: ACM, April 14–20 2024, pp. 1–13.
- [41] I. Jacobson, G. Booch, and J. Rumbaugh, *The Unified Software Development Process*. Addison-Wesley, 1999, vol. 1.
- [42] A. Jamil, M. Arif, N. Abubakar, and A. Ahmad, “Software testing techniques: A literature review,” 11 2016, pp. 177–182.
- [43] Z. Jiang, F. F. Xu, J. Araki, and G. Neubig, “How Can We Know What Language Models Know?” *Transactions of the Association for Computational Linguistics*, vol. 8, pp. 423–438, 07 2020. [Online]. Available: https://doi.org/10.1162/tacl_a_00324
- [44] D. Jurafsky and J. H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Pearson, 2009.
- [45] S. Kathiriya, R. Karangara, and N. Challa, “Optimizing automated software testing with machine learning techniques,” *International Journal of Science and Research (IJSR)*, vol. 7, pp. 2319–7064, 03 2018.
- [46] J. Kemper. (2023, March) Openai kills its codex code model, recommends gpt3.5 instead. Retrieved May 27, 2024. [Online]. Available: <https://thedeoder.com/openai-kills-its-codex-code-model-recommends-gpt3-5-instead/>
- [47] S. Kim, J. Zhao, Y. Tian, and S. Chandra, “Code prediction by feeding trees to transformers,” in *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 2021, pp. 150–162.
- [48] G. J. Klir and B. Yuan, *Fuzzy sets and fuzzy logic: theory and applications*. Prentice Hall, 1995.
- [49] H. H. Koester and S. Levine, “Effect of a word prediction feature on user performance,” *Augmentative and alternative communication*, vol. 12, no. 3, pp. 155–168, 1996.
- [50] Z. Liu, “Chatgpt will command more than 30,000 nvidia gpus: Report,” *Tom’s Hardware*, Mar 2023, archived from the original on May 30, 2024. Retrieved November 2, 2023.
- [51] S. Lu, D. Guo, S. Ren, J. Huang, A. Svyatkovskiy, A. Blanco, C. Clement, D. Drain, D. Jiang, D. Tang, G. Li, L. Zhou, L. Shou, L. Zhou, M. Tufano, M. Gong, M. Zhou, N. Duan, N. Sundaresan, S. K. Deng, S. Fu, and S. Liu, “Codexglue: A machine learning benchmark dataset for code understanding and generation,” in *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021.
- [52] C. Luo, “A report on automatic code completion,” 03 2017.

- [53] “React,” Meta. [Online]. Available: <https://react.dev/>
- [54] C. Metz and T. Mickle, “Openai completes deal that values the company at \$80 billion,” *The New York Times*, Feb 2024, retrieved May 30, 2024.
- [55] “TypeScript,” Microsoft. [Online]. Available: <https://www.typescriptlang.org/>
- [56] E. F. Miller, “Introduction to software testing technology,” in *Software Testing & Validation Techniques*. IEEE, 1981, pp. 4–16.
- [57] “Gecko,” Mozilla. [Online]. Available: <https://firefox-source-docs.mozilla.org/overview/gecko.html>
- [58] “Spidermonkey,” Mozilla. [Online]. Available: <https://spidermonkey.dev/>
- [59] A. H. Mughal, “Advancing bdd software testing: Dynamic scenario re-usability and step auto-complete for cucumber framework,” 2024. [Online]. Available: <https://arxiv.org/abs/2402.15928>
- [60] Netscape and Sun, “Netscape and sun announce javascript, the open, cross-platform object scripting language for enterprise networks and the internet,” Press release, 12 1995, retrieved June 01 2024.
- [61] “Nextauth,” NextAuth. [Online]. Available: <https://next-auth.js.org/>
- [62] C. Omar, Y. Yoon, T. LaToza, and B. Myers, “Active code completion,” in *2012 34th International Conference on Software Engineering (ICSE)*. IEEE, 2012, pp. 859–869.
- [63] OpenAI, “Chatgpt: Optimizing language models for dialogue,” 2022. [Online]. Available: <https://web.archive.org/web/20221130180912/https://openai.com/blog/chatgpt/>
- [64] “Jest,” OpenJS Foundation. [Online]. Available: <https://jestjs.io/>
- [65] “Node.js,” OpenJs Foundation. [Online]. Available: <https://nodejs.org>
- [66] “Mysql,” Oracle. [Online]. Available: <https://www.mysql.com/>
- [67] O. Parry, G. M. Kapfhammer, M. Hilton, and P. McMinn, “A survey of flaky tests,” *Transactions on Software Engineering and Methodology*, vol. 31, no. 1, 2022.
- [68] R. Patton, *Software Testing*. Indianapolis, IN: Sams Publishing, 2005.
- [69] “Prisma,” Prisma. [Online]. Available: <https://www.prisma.io/>
- [70] Prisma, “Many-to-many relations,” <https://www.prisma.io/docs/concepts/components/prisma-schema/relations/many-to-many-relations>, 2022, retrieved 2024-06-11.
- [71] J. K. Rask, F. P. Madsen, N. Battle, H. D. Macedo, and P. G. Larsen, “The specification language server protocol: A proposal for standardised lsp extensions,” 2022.
- [72] V. Raychev, M. Vechev, and E. Yahav, “Code completion with statistical language models,” in *Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation*. ACM, 2014, pp. 419–428.

- [73] R. Rodriguez-Echeverria, J. L. Canovas Izquierdo, M. Wimmer, and J. Cabot, “Towards a language server protocol infrastructure for graphical modeling,” in *Proceedings of the 21th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems*, ser. MODELS ’18. New York, NY, USA: Association for Computing Machinery, 2018, pp. 370–380.
- [74] T. J. Ross, *Fuzzy logic with engineering applications*. John Wiley & Sons, 2010.
- [75] E. Roth, “Microsoft spent hundreds of millions of dollars on a chatgpt supercomputer,” *The Verge*, Mar 2023, archived from the original on May 30, 2023. Retrieved March 30, 2023.
- [76] M. Shaw, “Prospects for an engineering discipline of software,” *IEEE Software*, pp. 15–24, Nov 1990.
- [77] C. Solis Pineda and X. Wang, “A study of the characteristics of behaviour driven development,” 10 2011, pp. 383 – 387.
- [78] Staff, “Using copilot chat in vs code,” *Visual Studio Code Documentation*, March 2024, retrieved June 20, 2024.
- [79] ——, “Github copilot – november 30th update,” *GitHub Blog*, November 2023, retrieved May 30, 2024. [Online]. Available: <https://github.blog/changelog/2023-11-30-github-copilot-november-30th-update/>
- [80] ——, “The top programming languages,” 2022, retrieved May 30, 2024. [Online]. Available: <https://octoverse.github.com/2022/top-programming-languages>
- [81] ——, “2022 stack overflow developer survey,” *Stack Overflow Blog*, June 2022, retrieved Jul 29, 2024.
- [82] ——, “2023 stack overflow developer survey,” *Stack Overflow Blog*, June 2023, retrieved Jul 29, 2024.
- [83] ——, “2024 stack overflow developer survey,” *Stack Overflow Blog*, May 2024, retrieved Jul 29, 2024.
- [84] K. Staff, “Microsoft-backed openai valued at \$80bn after company completes deal,” *The Guardian*, Feb 2024, retrieved May 30, 2024.
- [85] A. Svyatkovskiy, S. K. Deng, S. Fu, and N. Sundaresan, “Intellicode compose: Code generation using transformer,” in *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2020, pp. 1433–1443.
- [86] A. Svyatkovskoy, S. Lee, A. Hadjitofi, M. Riechert, J. Franco, and M. Allamanis, “Fast and memory-efficient neural code completion,” *arXiv preprint arXiv:2004.13651*, 2020.
- [87] “create-t3-app,” <https://github.com/t3-oss/create-t3-app>, T3 Open Source, 2022.
- [88] T. V. C. Team, “Visual studio code 1.0!” *Visual Studio Code Blog*, April 2016, retrieved June 1, 2024.

- [89] “trpc,” tRPC. [Online]. Available: <https://trpc.io/>
- [90] Unknown, “Tabnine: Coding in vs code with the help of an ai assistant,” learn.microsoft.com, March 2021, retrieved April 19, 2024.
- [91] “Next.js,” Vercel. [Online]. Available: <https://nextjs.org/>
- [92] J. Vincent, “This ai-powered autocomplete software is gmail’s smart compose for coders,” *The Verge*, July 2019, retrieved May 19, 2024.
- [93] Y. Wang and H. Li, “Code completion by modeling flattened abstract syntax trees as graphs,” in *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, 2021, pp. 14 015–14 023.
- [94] Y. Wang, W. Wang, S. Joty, and S. C. Hoi, “Codet5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation,” in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 2021.
- [95] B. Xu, A. Yang, J. Lin, Q. Wang, C. Zhou, Y. Zhang, and Z. Mao, “Expert-prompting: Instructing large language models to be distinguished experts,” 2023.
- [96] L. Zadeh, “Fuzzy sets,” *Information and Control*, vol. 8, no. 3, pp. 338–353, 1965. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S001999586590241X>
- [97] S. Zhao, “Github copilot chat now generally available for organizations and individuals,” *GitHub Blog*, December 2023, retrieved May 30, 2024. [Online]. Available: <https://github.blog/2023-12-29-github-copilot-chat-now-generally-available-for-organizations-and-individuals>
- [98] S. Z. Zhao, “Smarter, more efficient coding: Github copilot goes beyond codex with improved ai model,” *GitHub Blog*, July 2023, retrieved May 27, 2024. [Online]. Available: <https://github.blog/2023-07-28-smarter-more-efficient-coding-github-copilot-goes-beyond-codex-with-improv>
- [99] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong *et al.*, “A survey of large language models,” 2023.
- [100] Q. Zhou, C. Li, J. Yu, Y. Liu, G. Wang, K. Zhang, C. Ji, Q. Yan, L. He *et al.*, “A comprehensive survey on pretrained foundation models: A history from bert to chatgpt,” *arXiv preprint arXiv:2302.09419*, 2023.
- [101] X. Zhou, H. Zhu, L. Mathur, R. Zhang, H. Yu, Z. Qi, L.-P. Morency, Y. Bisk, D. Fried, G. Neubig, and M. Sap, “Sotopia: Interactive evaluation for social intelligence in language agents,” 2024.