Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Πληροφορική»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Ανάπτυξη Διαδικτυακής Πλατφόρμας Αξιολόγησης και Κατηγοριοποίησης Βιντεοπαιχνιδιών από Χρήστες** **Videogames Review and Categorization Social Platform**
Ονοματεπώνυμο Φοιτητή	**ΜΠΕΛΙΑΣ ΝΙΚΟΛΑΟΣ**
Πατρώνυμο	**ΔΗΜΗΤΡΙΟΣ**
Αριθμός Μητρώου	**ΜΠΠΛ2228**
Επιβλέπων	**Ευάγγελος Σακκόπουλος Αν. Καθηγητής**

Ημερομηνία Παράδοσης: Μάρτιος 2026

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)	(υπογραφή)	(υπογραφή)
Ιωάννης Βενέτης,	Ευάγγελος Σακκόπουλος	Ιωάννης Τασούλας,
Επικ. Καθηγητής	Αν. Καθηγητής (Επιβλέπων)	Επικ. Καθηγητής

Επιτελική Σύνοψη

Η παρούσα πτυχιακή εργασία επικεντρώνεται στην ανάπτυξη μιας πλήρως responsive διαδικτυακής πλατφόρμας αξιολόγησης, κατηγοριοποίησης και ανακάλυψης βιντεοπαιχνιδιών από χρήστες. Η πλατφόρμα στοχεύει να αποτελέσει ένα ενιαίο σημείο συγκέντρωσης και οργάνωσης πληροφοριών που προέρχονται από την κοινότητα των παικτών, καλύπτοντας το κενό που δημιουργείται από την ύπαρξη αποσπασματικών δεδομένων διάσπαρτων σε ιστοσελίδες, forums και online καταστήματα. Μέσα από τη συγκέντρωση αξιολογήσεων και σχολίων από τους ίδιους τους χρήστες, επιδιώκεται η δημιουργία ενός ολοκληρωμένου συστήματος υποστήριξης απόφασης και ανακάλυψης νέων τίτλων, με προσωποποιημένες προτάσεις παιχνιδιών.

Η πλατφόρμα προσφέρει ένα σύνολο λειτουργιών που καλύπτουν όλο το φάσμα αλληλεπίδρασης του χρήστη με το περιεχόμενο. Περιλαμβάνει σύστημα εγγραφής και σύνδεσης μέσω Google Authentication, δυνατότητα διαχείρισης προφίλ και πρόσβαση σε εκτενείς λίστες βιντεοπαιχνιδιών. Κάθε παιχνίδι συνοδεύεται από λεπτομερή σελίδα πληροφοριών, όπως περιγραφή, είδος, έτος κυκλοφορίας, developers, publishers, game modes και μέση βαθμολογία που υπολογίζεται δυναμικά. Οι χρήστες μπορούν να αξιολογήσουν παιχνίδια με κλίμακα αστεριών, να γράψουν σχόλια, να δημιουργήσουν και να διαχειριστούν collections, καθώς και να φιλτράρουν το περιεχόμενο σύμφωνα με τα ενδιαφέροντά τους.

Επιπλέον, η εργασία ενσωματώνει ένα βασικό σύστημα προτάσεων (recommendation system), το οποίο λειτουργεί σε επίπεδο collection και αξιοποιεί δύο προσεγγίσεις: **content-based filtering**, που βασίζεται στα χαρακτηριστικά των παιχνιδιών που έχει προσθέσει ο χρήστης, και **collaborative filtering**, που αντλεί προτεινόμενους τίτλους από collections άλλων χρηστών με παρόμοια προτίμηση. Το σύστημα συμβάλλει στη βελτίωση της εμπειρίας αναζήτησης, προσφέροντας πιο στοχευμένες και σχετικές προτάσεις.

Για την υλοποίηση της πλατφόρμας χρησιμοποιήθηκε σύγχρονο τεχνολογικό stack. Το frontend αναπτύχθηκε με **Next.js** σε **TypeScript**, με χρήση **React**, **TailwindCSS** και στοιχείων **shadcn/ui**, ενώ η επικοινωνία με το backend υλοποιήθηκε μέσω **axios** και **React Query**. Το backend βασίστηκε στο **Django Rest Framework**, προσφέροντας ένα ασφαλές και επεκτάσιμο REST API. Η **PostgreSQL** αποτέλεσε τη βάση δεδομένων για την αποθήκευση χρηστών, παιχνιδιών, αξιολογήσεων, genres, tags, platforms και collections. Τέλος, χρησιμοποιήθηκε εξωτερική πηγή δεδομένων μέσω IGDB API για την αρχική εισαγωγή παιχνιδιών.

Η διαδικασία ανάπτυξης ακολούθησε μια επαναληπτική μεθοδολογία, όπου αρχικά υλοποιήθηκαν ο πυρήνας των λειτουργιών (authentication, προβολή παιχνιδιών), στη συνέχεια ο μηχανισμός αξιολόγησης και αργότερα η κατηγοριοποίηση, τα φίλτρα και οι προτάσεις. Σε κάθε στάδιο πραγματοποιήθηκαν λειτουργικές δοκιμές, διορθώσεις προβλημάτων και προσαρμογές στο UI. Επιπλέον, πραγματοποιήθηκαν βασικά tests χρηστικότητας με συμμετοχή χρηστών, που αξιολόγησαν την αναζήτηση παιχνιδιών, τη χρήση φίλτρων και τη διαδικασία αξιολόγησης. Το σύστημα προτάσεων δοκιμάστηκε ώστε να διασφαλιστεί η συνοχή και η καταλληλότητα των αποτελεσμάτων του.

Τα αποτελέσματα της εργασίας περιλαμβάνουν μια ολοκληρωμένη και λειτουργική web εφαρμογή, ικανή να εξυπηρετήσει όλες τις βασικές ανάγκες ενός χρήστη που θέλει να ανακαλύψει, να αξιολογήσει και να οργανώσει βιντεοπαιχνίδια. Η πλατφόρμα έχει υλοποιηθεί σε παραγωγικό επίπεδο, με σύγχρονο responsive design και υψηλή ευχρηστία. Τα σχόλια των χρηστών επιβεβαιώνουν ότι η διεπαφή είναι εύχρηστη και οι προτάσεις παιχνιδιών χρήσιμες, ενώ οι χρόνοι απόκρισης κρίνονται ικανοποιητικοί.

Συνολικά, η εργασία αποδεικνύει πως η συλλογική γνώση της κοινότητας μπορεί να αξιοποιηθεί για τη διαμόρφωση ενός ευέλικτου συστήματος ανακάλυψης παιχνιδιών, το οποίο οργανώνει αποτελεσματικά πληροφορίες και παρέχει προσωποποιημένες εμπειρίες. Παρά τα όρια του τρέχοντος συστήματος προτάσεων, το οποίο βασίζεται σε απλούστερους αλγορίθμους και διαθέτει περιορισμένο όγκο δεδομένων, η πλατφόρμα θέτει ισχυρές βάσεις για μελλοντικές επεκτάσεις όπως εφαρμογή machine learning, ενσωμάτωση κοινωνικών λειτουργιών και ανάπτυξη mobile έκδοσης.

Ευχαριστίες

Θα ήθελα να εκφράσω την ειλικρινή μου ευγνωμοσύνη σε όλους όσους με στήριξαν κατά τη διάρκεια της εκπόνησης της πτυχιακής μου εργασίας, αλλά και σε όλη την πορεία των σπουδών μου.

Πρώτα απ' όλα, ευχαριστώ θερμά τους γονείς μου, Δημήτρη και Αργυρώ, και τον αδερφό μου Γιώργο, για την αδιάκοπη συμπαράσταση, την υπομονή και την αγάπη τους. Χωρίς τη δύναμη και την εμπιστοσύνη που μου έδωσαν όλα αυτά τα χρόνια, δεν θα είχα φτάσει μέχρι εδώ.

Επιπλέον, θα ήθελα να ευχαριστήσω από καρδιάς τη σύζυγό μου, Βασιλική, για την αμέριστη στήριξη, την κατανόηση και την ενθάρρυνσή της σε όλη αυτή τη διαδρομή. Η παρουσία της αποτέλεσε για μένα σταθερό σημείο αναφοράς, έμπνευση και πηγή ψυχικής δύναμης.

Ένα μεγάλο ευχαριστώ και σε όλη την οικογένειά μου, για την ηθική υποστήριξη και το ενδιαφέρον τους σε κάθε μου βήμα.

Θα ήθελα επίσης να ευχαριστήσω τους αγαπημένους μου φίλους, Δημήτρη, Βασίλη, Χαράλαμπο και Θωμά, που βρίσκονταν πάντα δίπλα μου, είτε για να με ενθαρρύνουν, είτε για να μου θυμίζουν με τον τρόπο τους ότι κάθε δυσκολία ξεπερνιέται. Η παρουσία τους αποτέλεσε σημαντική πηγή δύναμης και ισορροπίας.

Ιδιαίτερη μνεία αξίζει ο επιβλέπων της εργασίας μου, Αναπληρωτής Καθηγητής Ε. Σακκόπουλος, για την καθοδήγηση, τις εύστοχες επισημάνσεις και την πολύτιμη βοήθειά του καθ' όλη τη διάρκεια της εκπόνησης. Η συμβολή του υπήρξε καθοριστική για την ολοκλήρωση της παρούσας εργασίας.

Σε όλους όσους συνέβαλαν με τον τρόπο τους σ' αυτό το ταξίδι, μικρό ή μεγάλο, εκφράζω την ευγνωμοσύνη μου.

ΙΑΝΟΥΆΡΙΟΣ 2026

ΜΠΕΛΙΑΣ ΝΙΚΟΛΑΟΣ

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

Κεφάλαιο 1ο

Εισαγωγή

Η ραγδαία ανάπτυξη της βιομηχανίας των βιντεοπαιχνιδιών, σε συνδυασμό με την εξάπλωση των διαδικτυακών πλατφορμών και των εργαλείων διαμοιρασμού περιεχομένου, έχει δημιουργήσει νέες ανάγκες για οργάνωση, αξιολόγηση και εξατομικευμένη παρουσίαση πληροφορίας. Οι παίκτες έρχονται καθημερινά σε επαφή με ένα τεράστιο πλήθος τίτλων, ειδών και πλατφορμών, γεγονός που καθιστά δύσκολη την εύρεση αξιόπιστων και στοχευμένων προτάσεων. Ταυτόχρονα, η γνώση που παράγεται από τις κοινότητες των χρηστών (crowdsourcing) αποτελεί πολύτιμη πηγή πληροφοριών, η οποία, όταν οργανωθεί σωστά, μπορεί να οδηγήσει σε πιο αποτελεσματικούς μηχανισμούς αναζήτησης και σύστασης περιεχομένου.

Η παρούσα εργασία εντάσσεται στο επιστημονικό πεδίο της ανάπτυξης διαδικτυακών πληροφοριακών συστημάτων, με έμφαση στις τεχνολογίες web εφαρμογών, στα συστήματα διαχείρισης και αξιολόγησης περιεχομένου, καθώς και στις μεθοδολογίες σύστασης (recommendation systems). Η προσέγγιση είναι κυρίως πρακτική, καθώς περιλαμβάνει την ανάλυση, τον σχεδιασμό και την υλοποίηση μιας πλήρως λειτουργικής διαδικτυακής πλατφόρμας, αξιοποιώντας σύγχρονα εργαλεία ανάπτυξης όπως Next.js, Django Rest Framework και PostgreSQL.

1.1 Περιγραφή του υπό μελέτη προβλήματος

Το πρόβλημα που εξετάζεται αφορά την έλλειψη ενός ενιαίου, καλά οργανωμένου και διαδραστικού χώρου όπου οι χρήστες μπορούν να αξιολογούν και να κατηγοριοποιούν βιντεοπαιχνίδια και παράλληλα να λαμβάνουν εξατομικευμένες προτάσεις. Στη σημερινή πραγματικότητα, η πληροφορία σχετικά με τα βιντεοπαιχνίδια είναι διάσπαρτη σε πολλά sites, forums και ηλεκτρονικά καταστήματα, χωρίς να υπάρχει ένας ολοκληρωμένος μηχανισμός συγκέντρωσης και αξιοποίησης των απόψεων των χρηστών. Αυτό δυσχεραίνει την ανακάλυψη νέων τίτλων από τους παίκτες και περιορίζει τη δυνατότητα διαμόρφωσης ενός προσωπικού προφίλ προτιμήσεων.

1.2 Σκοπός και στόχοι της εργασίας

Σκοπός της παρούσας εργασίας είναι η ανάπτυξη μιας responsive διαδικτυακής πλατφόρμας που επιτρέπει στους χρήστες να ανακαλύπτουν, να αξιολογούν και να οργανώνουν βιντεοπαιχνίδια, ενώ παράλληλα τους παρέχει προτάσεις βασισμένες στα ενδιαφέροντά τους.

Οι επιμέρους στόχοι περιλαμβάνουν:

- Την ανάλυση και αξιοποίηση σύγχρονων τεχνολογιών ανάπτυξης εξατομικευμένου λογισμικού.
- Τον σχεδιασμό και την υλοποίηση ενός συστήματος αξιολόγησης, κατηγοριοποίησης και αναζήτησης βιντεοπαιχνιδιών.
- Τη δημιουργία ενός βασικού recommendation system, με συνδυασμό content-based και collaborative filtering.
- Τη συγκριτική αξιολόγηση των λειτουργιών και της αποτελεσματικότητας του συστήματος προτάσεων.
- Την ανάπτυξη ενός φιλικού προς τον χρήστη περιβάλλοντος, εύχρηστου και κατάλληλου για πολλαπλές συσκευές.

1.3 Βασικοί ορισμοί

Στην ενότητα αυτή παρουσιάζονται οι θεμελιώδεις έννοιες που χρησιμοποιούνται στη συνέχεια της εργασίας, ώστε να υπάρχει κοινό εννοιολογικό πλαίσιο.

1.3.1 Recommendation Systems (Συστήματα Προτάσεων)

Πρόκειται για μεθοδολογίες και αλγοριθμικές τεχνικές που χρησιμοποιούνται για την παροχή εξατομικευμένων προτάσεων σε χρήστες. Οι δύο κύριες προσεγγίσεις που αξιοποιούνται στην παρούσα εργασία είναι:

- **Content-based filtering:** βασίζεται σε χαρακτηριστικά των αντικειμένων (π.χ. είδος παιχνιδιού, tags).
- **Collaborative filtering:** βασίζεται σε ομοιότητες μεταξύ χρηστών ή συλλογών.

1.3.2 Crowdsourcing

Η συλλογική συνεισφορά χρηστών που δημιουργεί γνώση και πληροφορίες μέσω αξιολογήσεων, σχολίων και κατηγοριοποίησης.

1.3.3 Web Application Architecture

Αναφέρεται στη δομή και λειτουργία μιας διαδικτυακής εφαρμογής, περιλαμβάνοντας backend, frontend και βάση δεδομένων.

1.3.4 Χρησιμοποιούμενες τεχνολογίες

Next.js / React / TypeScript: για την κατασκευή του περιβάλλοντος χρήστη.
Django Rest Framework: για την υλοποίηση του API και τη διαχείριση δεδομένων.
PostgreSQL: για την αποθήκευση των πληροφοριών των χρηστών και των παιχνιδιών.
TailwindCSS και shadcn/ui: για την εμφάνιση και το responsive design.

1.3.5 User Generated Content

Περιεχόμενο που δημιουργείται από τους ίδιους τους χρήστες μιας πλατφόρμας, όπως αξιολογήσεις, σχόλια, λίστες και συλλογές, και αποτελεί βασικό στοιχείο συμμετοχικών web εφαρμογών.

1.3.6 Cold Start Problem

Πρόβλημα που εμφανίζεται στα συστήματα προτάσεων όταν δεν υπάρχουν επαρκή δεδομένα για νέους χρήστες ή νέα αντικείμενα, με αποτέλεσμα μειωμένη ακρίβεια στις προτάσεις.

1.4 Παραδοτέα της εργασίας

Τα παραδοτέα της παρούσας πτυχιακής περιλαμβάνουν:

- Το πλήρες έντυπο κείμενο της εργασίας, το οποίο παρουσιάζει τη θεωρητική ανάλυση, τη μεθοδολογία υλοποίησης και τα αποτελέσματα.
- Τη λειτουργική web εφαρμογή που υλοποιήθηκε, μαζί με τον πηγαίο κώδικα, τις διαμορφώσεις και τη βάση δεδομένων.
- Το σύνολο των πηγών και της βιβλιογραφίας που συλλέχθηκαν, αποτελώντας μια μικρή βάση γνώσης σχετική με τα συστήματα αξιολόγησης και προτάσεων.

1.5 Δομή της εργασίας

Η εργασία οργανώνεται ως εξής:

- **Κεφάλαιο 1 – Εισαγωγή:** παρουσιάζει το πλαίσιο, το πρόβλημα, τους στόχους και τα παραδοτέα.
- **Κεφάλαιο 2 – Θεωρητικό υπόβαθρο:** περιγράφει τις τεχνολογίες web development, τα συστήματα προτάσεων και συναφείς επιστημονικές έννοιες.
- **Κεφάλαιο 3 – Ανάλυση απαιτήσεων και σχεδιασμός:** περιλαμβάνει use cases, μοντέλα δεδομένων και αρχιτεκτονική.
- **Κεφάλαιο 4 – Υλοποίηση της πλατφόρμας:** αναλύει λεπτομερώς την ανάπτυξη του frontend, backend και της βάσης δεδομένων.
- **Κεφάλαιο 5 – Αξιολόγηση και αποτελέσματα:** παρουσιάζει τα tests, τις μετρήσεις και την απόδοση του recommendation system.
- **Κεφάλαιο 6 – Συμπεράσματα και μελλοντικές επεκτάσεις:** συνοψίζει τα ευρήματα και προτείνει βελτιώσεις για τη συνέχεια.

Κεφάλαιο 2ο

Επισκόπηση του χώρου

Η παρούσα εργασία εντάσσεται στον ευρύτερο χώρο των διαδικτυακών πληροφοριακών συστημάτων και των πλατφορμών διαχείρισης και αξιολόγησης ψηφιακού περιεχομένου, με ειδικότερη εστίαση στα βιντεοπαιχνίδια. Στο κεφάλαιο αυτό παρουσιάζεται η επισκόπηση του χώρου που σχετίζεται με την ανάπτυξη πλατφορμών αξιολόγησης περιεχομένου από χρήστες, τα συστήματα κατηγοριοποίησης και τα συστήματα προτάσεων, καθώς και οι σύγχρονες τεχνολογικές τάσεις στο web development που υποστηρίζουν τέτοιες εφαρμογές.

2.1 Πλατφόρμες αξιολόγησης και κατηγοριοποίησης περιεχομένου

Οι πλατφόρμες αξιολόγησης περιεχομένου αποτελούν ένα από τα πιο διαδεδομένα παραδείγματα εφαρμογών Web 2.0, όπου οι χρήστες δεν είναι απλοί καταναλωτές πληροφορίας, αλλά ενεργοί δημιουργοί περιεχομένου. Μέσω μηχανισμών όπως βαθμολογίες, σχόλια, ετικέτες (tags) και λίστες, οι χρήστες συμβάλλουν στη διαμόρφωση μιας συλλογικής βάσης γνώσης.
Χαρακτηριστικά παραδείγματα τέτοιων πλατφορμών στον χώρο της ψυχαγωγίας είναι ιστοσελίδες αξιολόγησης ταινιών, σειρών, μουσικής και βιντεοπαιχνιδιών. Οι πλατφόρμες αυτές επιτρέπουν:

- την αποθήκευση και οργάνωση μεγάλου όγκου περιεχομένου,
- την κατηγοριοποίηση βάσει ειδών, θεματικών ή χαρακτηριστικών,
- τη συγκέντρωση αξιολογήσεων από διαφορετικούς χρήστες,
- και τη σύγκριση αντικειμένων βάσει στατιστικών δεδομένων.

Στον τομέα των βιντεοπαιχνιδιών, η ανάγκη για τέτοιες πλατφόρμες είναι ιδιαίτερα έντονη, καθώς ο αριθμός των διαθέσιμων τίτλων αυξάνεται ραγδαία και οι πληροφορίες συχνά προέρχονται από ετερογενείς και αποσπασματικές πηγές.

2.2 Συμμετοχικά συστήματα και crowdsourcing

Η έννοια του crowdsourcing αναφέρεται στη συλλογική συνεισφορά ενός μεγάλου αριθμού χρηστών για τη δημιουργία, αξιολόγηση ή βελτίωση περιεχομένου. Στις πλατφόρμες αξιολόγησης βιντεοπαιχνιδιών, το crowdsourcing εκφράζεται μέσω:

- βαθμολογιών,
- σχολίων και κριτικών,
- δημιουργίας συλλογών (collections),
- και προσθήκης μεταδεδομένων όπως genres και tags.

Η αξιοποίηση της συλλογικής γνώσης επιτρέπει την παραγωγή πιο αξιόπιστων και αντιπροσωπευτικών αποτελεσμάτων σε σύγκριση με μεμονωμένες απόψεις. Παράλληλα,

δημιουργεί ένα δυναμικό σύστημα που εξελίσσεται συνεχώς όσο αυξάνεται η συμμετοχή των χρηστών.
Ωστόσο, τα συμμετοχικά συστήματα αντιμετωπίζουν και προκλήσεις, όπως η ποιότητα των δεδομένων, η μεροληψία στις αξιολογήσεις και η ανάγκη για μηχανισμούς φιλτραρίσματος και κανονικοποίησης της πληροφορίας.

2.3 Συστήματα προτάσεων (Recommendation Systems)

Τα συστήματα προτάσεων αποτελούν βασικό στοιχείο σύγχρονων ψηφιακών πλατφορμών, καθώς συμβάλλουν στην εξατομίκευση της εμπειρίας του χρήστη και στη μείωση της υπερφόρτωσης πληροφορίας. Στον χώρο των βιντεοπαιχνιδιών, τα recommendation systems βοηθούν τους χρήστες να ανακαλύψουν νέους τίτλους που ταιριάζουν στις προτιμήσεις τους.
Οι βασικές κατηγορίες συστημάτων προτάσεων περιλαμβάνουν:

- Content-based filtering, όπου οι προτάσεις βασίζονται στα χαρακτηριστικά των αντικειμένων που έχει ήδη επιλέξει ή αξιολογήσει ο χρήστης (π.χ. είδος παιχνιδιού, tags, game modes).
- Collaborative filtering, όπου οι προτάσεις προκύπτουν από τη σύγκριση προτιμήσεων μεταξύ διαφορετικών χρηστών ή συλλογών με παρόμοια χαρακτηριστικά.
- Υβριδικά συστήματα, τα οποία συνδυάζουν τις παραπάνω προσεγγίσεις για βελτιωμένα αποτελέσματα.

Η παρούσα εργασία υιοθετεί έναν συνδυασμό content-based και collaborative filtering, εστιάζοντας στη λειτουργία των collections ως βασικό σημείο σύγκλισης των προτιμήσεων των χρηστών.

2.4 Τεχνολογίες ανάπτυξης σύγχρονων web εφαρμογών

Η ανάπτυξη σύγχρονων διαδικτυακών πλατφορμών βασίζεται σε αρχιτεκτονικές που διαχωρίζουν το frontend από το backend, προσφέροντας ευελιξία, επεκτασιμότητα και καλύτερη εμπειρία χρήστη.

Στο frontend, κυριαρχούν frameworks βασισμένα στη JavaScript, όπως το React και το Next.js, τα οποία επιτρέπουν τη δημιουργία δυναμικών και responsive διεπαφών χρήστη. Η χρήση τεχνολογιών όπως το TypeScript συμβάλλει στη βελτίωση της ποιότητας και της συντηρησιμότητας του κώδικα.

Στο backend, RESTful APIs υλοποιούνται συχνά με frameworks όπως το Django Rest Framework, παρέχοντας μηχανισμούς ασφάλειας, διαχείρισης χρηστών και πρόσβασης σε δεδομένα. Η αποθήκευση δεδομένων πραγματοποιείται συνήθως σε σχεσιακές βάσεις δεδομένων, όπως η PostgreSQL, οι οποίες προσφέρουν αξιοπιστία και ισχυρές δυνατότητες μοντελοποίησης.

2.5 Υφιστάμενες πλατφόρμες αξιολόγησης βιντεοπαιχνιδιών

Στον χώρο των βιντεοπαιχνιδιών υπάρχουν ήδη διαδικτυακές πλατφόρμες που παρέχουν πληροφορίες, αξιολογήσεις και μηχανισμούς ανακάλυψης περιεχομένου. Οι πλατφόρμες αυτές διαφέρουν ως προς τον βαθμό συμμετοχής των χρηστών, τον τρόπο οργάνωσης της πληροφορίας και το επίπεδο εξατομίκευσης που προσφέρουν.

Μία ευρέως γνωστή κατηγορία πλατφορμών είναι τα εξειδικευμένα portals πληροφόρησης και κριτικών, όπου η αξιολόγηση των παιχνιδιών βασίζεται κυρίως σε επαγγελματικές κριτικές και βαθμολογίες. Αν και οι πλατφόρμες αυτές προσφέρουν αξιόπιστη και δομημένη πληροφορία, συχνά περιορίζουν τη συμμετοχή των απλών χρηστών και δεν υποστηρίζουν σε μεγάλο βαθμό την εξατομίκευση βάσει προσωπικών προτιμήσεων.

Παράλληλα, υπάρχουν πλατφόρμες που λειτουργούν ως ψηφιακά καταστήματα και κοινότητες χρηστών, επιτρέποντας την υποβολή αξιολογήσεων και σχολίων από παίκτες. Οι πλατφόρμες αυτές διαθέτουν μεγάλο όγκο δεδομένων και ενεργές κοινότητες, ωστόσο η πληροφορία είναι συχνά συνδεδεμένη με εμπορικούς σκοπούς, γεγονός που μπορεί να επηρεάσει την ουδετερότητα της παρουσίασης και τη δομή των προτάσεων.

Επιπλέον, συναντώνται βάσεις δεδομένων παιχνιδιών με έμφαση στην τεκμηρίωση και στα μεταδεδομένα (genres, platforms, developers). Αν και αποτελούν πολύτιμες πηγές πληροφορίας, συνήθως δεν ενσωματώνουν εξελιγμένους μηχανισμούς κοινωνικής αλληλεπίδρασης ή εξατομικευμένες προτάσεις περιεχομένου.

Η πλατφόρμα που αναπτύσσεται στο πλαίσιο της παρούσας εργασίας διαφοροποιείται, καθώς εστιάζει στη συνδυαστική αξιοποίηση της συλλογικής γνώσης των χρηστών, της κατηγοριοποίησης μέσω collections και της παροχής εξατομικευμένων προτάσεων. Στόχος δεν είναι η αντικατάσταση υφιστάμενων πλατφορμών, αλλά η παροχή ενός ενοποιημένου και καθαρά συμμετοχικού περιβάλλοντος, προσανατολισμένου στην εμπειρία του χρήστη και όχι σε εμπορικούς μηχανισμούς.

Η αξιοποίηση εξωτερικών APIs, όπως βάσεις δεδομένων παιχνιδιών, αποτελεί συνήθη πρακτική για τον αρχικό εμπλουτισμό του περιεχομένου και τη μείωση του κόστους συλλογής δεδομένων.

2.6 Προκλήσεις και περιορισμοί στα συστήματα αξιολόγησης και προτάσεων

Παρά τα σημαντικά πλεονεκτήματα που προσφέρουν τα συστήματα αξιολόγησης και προτάσεων, η ανάπτυξη και λειτουργία τους συνοδεύεται από μια σειρά προκλήσεων και περιορισμών που επηρεάζουν την αποτελεσματικότητά τους.

Μία βασική πρόκληση είναι το λεγόμενο cold start problem, το οποίο εμφανίζεται όταν νέοι χρήστες ή νέα αντικείμενα εισέρχονται στο σύστημα χωρίς να διαθέτουν επαρκή ιστορικά δεδομένα. Σε τέτοιες περιπτώσεις, η παραγωγή αξιόπιστων προτάσεων καθίσταται δύσκολη, ιδιαίτερα για τα collaborative filtering συστήματα.

Επιπλέον, η υποκειμενικότητα των αξιολογήσεων αποτελεί έναν σημαντικό περιορισμό. Οι βαθμολογίες και τα σχόλια των χρηστών επηρεάζονται από προσωπικές προτιμήσεις, εμπειρίες και συγκυριακούς παράγοντες, γεγονός που μπορεί να οδηγήσει σε μεροληπτικά ή μη αντιπροσωπευτικά αποτελέσματα.

Ένας ακόμη παράγοντας είναι ο περιορισμένος όγκος δεδομένων, ειδικά σε νέες ή μικρότερης κλίμακας πλατφόρμες. Ο μικρός αριθμός χρηστών και αξιολογήσεων μειώνει την ακρίβεια των συστημάτων προτάσεων και περιορίζει τις δυνατότητες εξαγωγής στατιστικά σημαντικών συμπερασμάτων.

Τέλος, σημαντική πρόκληση αποτελεί και η ισορροπία μεταξύ απλότητας και πολυπλοκότητας. Η χρήση πιο σύνθετων αλγορίθμων μπορεί να βελτιώσει την ποιότητα των προτάσεων, αυξάνει όμως το υπολογιστικό κόστος και την πολυπλοκότητα υλοποίησης. Για τον λόγο αυτό, σε πολλές περιπτώσεις επιλέγονται απλούστερες προσεγγίσεις που προσφέρουν ικανοποιητικά αποτελέσματα με μικρότερο κόστος.

Οι παραπάνω περιορισμοί λαμβάνονται υπόψη στην παρούσα εργασία, επηρεάζοντας τις σχεδιαστικές αποφάσεις και τον τρόπο υλοποίησης του συστήματος προτάσεων. Παρά τους περιορισμούς, η επιλεγμένη προσέγγιση κρίνεται κατάλληλη για το εύρος και τους στόχους της εφαρμογής, ενώ αφήνει περιθώρια για μελλοντικές επεκτάσεις και βελτιώσεις.

2.7 Σύνοψη κεφαλαίου

Στο κεφάλαιο αυτό παρουσιάστηκε η επισκόπηση του χώρου στον οποίο εντάσσεται η παρούσα εργασία, καλύπτοντας τις πλατφόρμες αξιολόγησης περιεχομένου, τα συμμετοχικά συστήματα, τα συστήματα προτάσεων και τις σύγχρονες τεχνολογίες web development. Το θεωρητικό αυτό υπόβαθρο αποτελεί τη βάση για την ανάλυση απαιτήσεων, τον σχεδιασμό και την υλοποίηση της διαδικτυακής πλατφόρμας που παρουσιάζεται στα επόμενα κεφάλαια, καθώς και για την αξιολόγηση των επιλογών που πραγματοποιήθηκαν κατά την ανάπτυξή της.

Κεφάλαιο 3ο

Εγχειρίδιο Χρήσης και Εγκατάστασης Πλατφόρμας

Το παρόν κεφάλαιο λειτουργεί ως εγχειρίδιο χρήσης και εγκατάστασης της διαδικτυακής πλατφόρμας αξιολόγησης και κατηγοριοποίησης βιντεοπαιχνιδιών που αναπτύχθηκε στο πλαίσιο της παρούσας εργασίας. Στόχος του είναι να παρουσιάσει το τελικό αποτέλεσμα της υλοποίησης, τη βασική ροή χρήσης από την πλευρά του τελικού χρήστη, καθώς και τα απαραίτητα βήματα για την τοπική εκτέλεση της εφαρμογής. Η περιγραφή γίνεται σε υψηλό επίπεδο, ώστε ο αναγνώστης να αποκτήσει σαφή εικόνα της λειτουργικότητας του συστήματος χωρίς να απαιτείται εις βάθος γνώση του πηγαίου κώδικα.

3.1 Περιγραφή του τελικού συστήματος

Η αναπτυγμένη πλατφόρμα αποτελεί μια πλήρως responsive web εφαρμογή, μέσω της οποίας οι χρήστες μπορούν να ανακαλύπτουν, να αξιολογούν και να οργανώνουν βιντεοπαιχνίδια. Το σύστημα βασίζεται σε αρχιτεκτονική διαχωρισμού frontend και backend, με το frontend να παρέχει το περιβάλλον αλληλεπίδρασης του χρήστη και το backend να διαχειρίζεται τα δεδομένα, την αυθεντικοποίηση και τη λογική των προτάσεων.

Η πλατφόρμα απευθύνεται σε χρήστες που επιθυμούν να συγκεντρώνουν πληροφορίες για βιντεοπαιχνίδια, να καταγράφουν τις προτιμήσεις τους και να λαμβάνουν προτάσεις βασισμένες στη δραστηριότητά τους και στις συλλογές άλλων χρηστών.

3.2 Ρόλοι χρηστών και παρεχόμενες λειτουργίες

Στο σύστημα διακρίνονται δύο βασικοί ρόλοι χρηστών:

Επισκέπτης (μη εγγεγραμμένος χρήστης):

- Προβολή της λίστας βιντεοπαιχνιδιών.
- Αναζήτηση και φιλτράρισμα περιεχομένου.
- Πρόσβαση σε βασικές πληροφορίες παιχνιδιών και στη μέση βαθμολογία τους.

Εγγεγραμμένος χρήστης:

- Δημιουργία και διαχείριση προσωπικού προφίλ.
- Αξιολόγηση παιχνιδιών μέσω κλίμακας αστεριών.
- Υποβολή σχολίων.
- Δημιουργία και διαχείριση collections.
- Λήψη προσωποποιημένων προτάσεων παιχνιδιών.

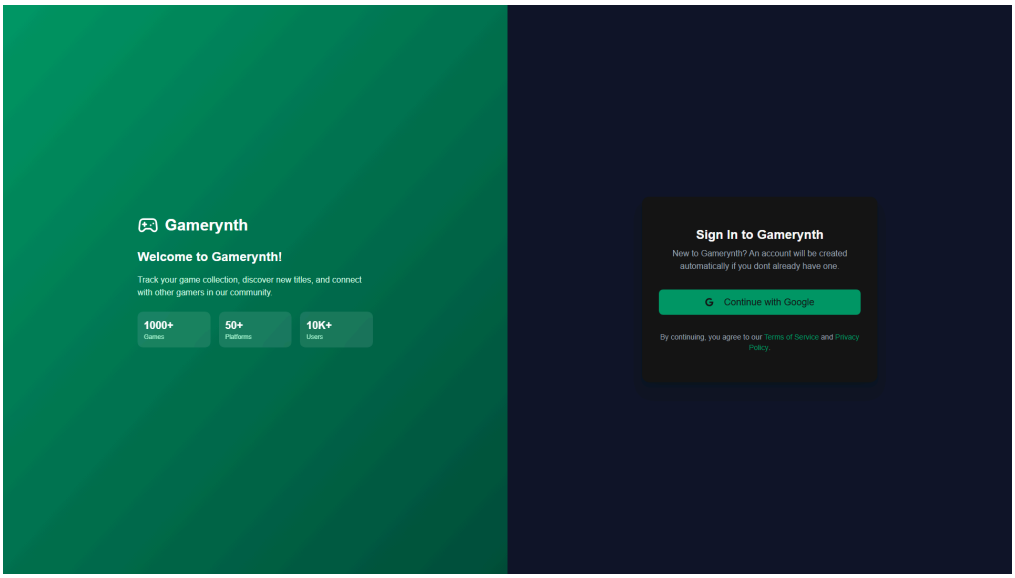
3.3 Εγχειρίδιο χρήσης της πλατφόρμας

3.3.1 Σελίδα σύνδεσης χρήστη

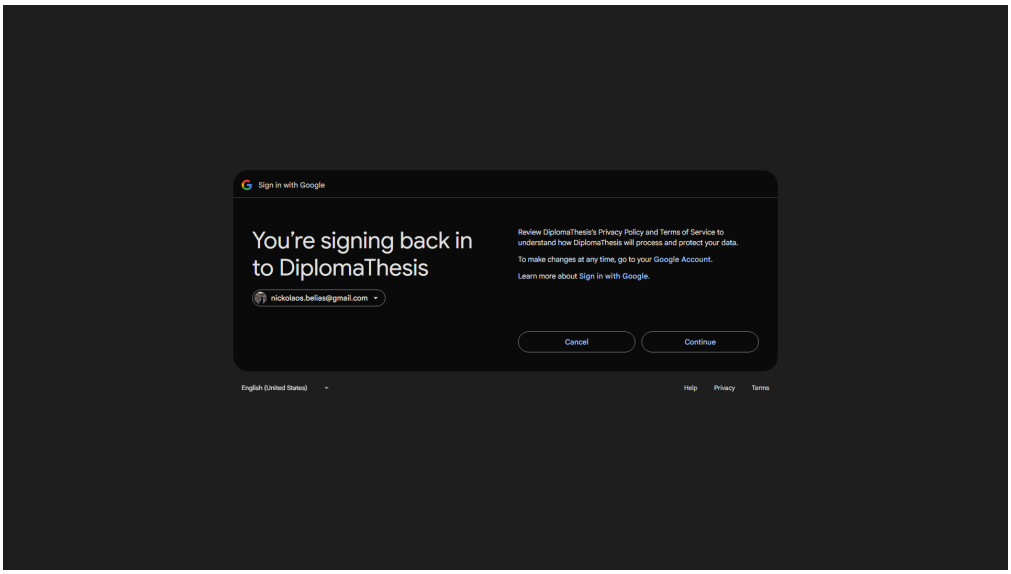
Η σελίδα αυτή λειτουργεί ως σημείο ελέγχου πρόσβασης και διασφαλίζει ότι οι χρήστες που αλληλεπιδρούν με τις πλήρεις λειτουργίες του συστήματος είναι αυθεντικοποιημένοι.

Η αυθεντικοποίηση πραγματοποιείται μέσω του μηχανισμού Google Authentication, αξιοποιώντας εξωτερική υπηρεσία ταυτοποίησης. Ο χρήστης έχει τη δυνατότητα να συνδεθεί με τον υπάρχοντα λογαριασμό Google που διαθέτει, χωρίς να απαιτείται η δημιουργία νέου ονόματος χρήστη και κωδικού πρόσβασης εντός της πλατφόρμας. Με τον τρόπο αυτό βελτιώνεται η εμπειρία χρήσης, ενώ παράλληλα ενισχύεται το επίπεδο ασφάλειας της εφαρμογής.

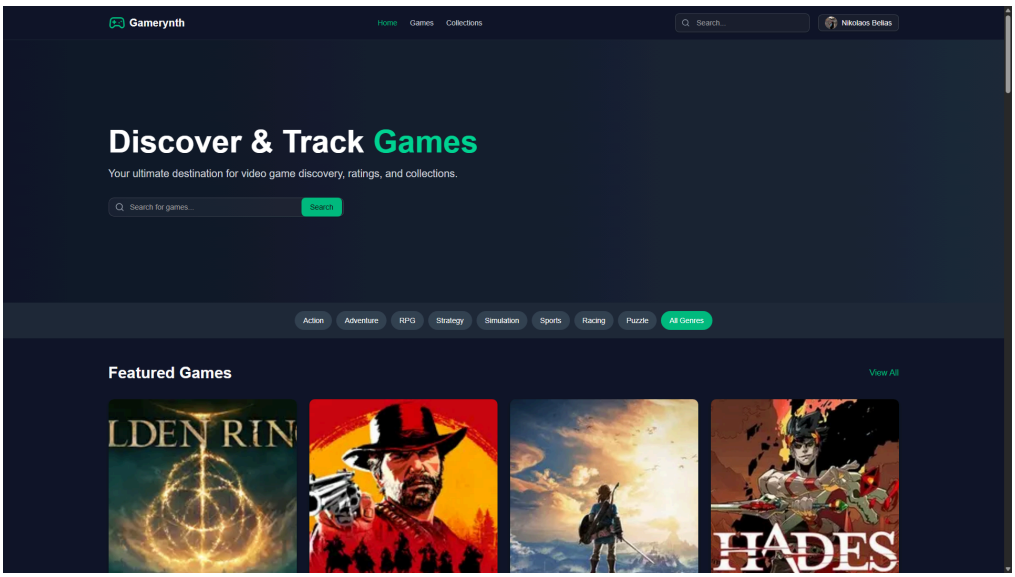
Μετά την επιτυχή ολοκλήρωση της διαδικασίας σύνδεσης, ο χρήστης ανακατευθύνεται αυτόματα στην κύρια σελίδα της πλατφόρμας. Εκεί μπορεί να επιβεβαιώσει ότι έχει συνδεθεί επιτυχώς, καθώς εμφανίζεται το προφίλ του πάνω δεξιά στο header, και αποκτά πρόσβαση στις διαθέσιμες λειτουργίες, όπως η αξιολόγηση παιχνιδιών, η δημιουργία collections και η λήψη εξατομικευμένων προτάσεων.



Εικόνα 1 Σελίδα σύνδεσης χρήστη



Εικόνα 2 Σελίδα επιλογής λογαριασμού Google



Εικόνα 3 Αρχική σελίδα εφαρμογής μετά την αυθεντικοποίηση του χρήστη

3.3.2 Αρχική σελίδα

Η αρχική σελίδα της πλατφόρμας λειτουργεί ως το κεντρικό σημείο πλοήγησης και παρέχει στον χρήστη πρόσβαση στις βασικές λειτουργίες της εφαρμογής. Μέσω αυτής, ο χρήστης μπορεί να πραγματοποιήσει αναζήτηση πληκτρολογώντας τον τίτλο του επιθυμητού παιχνιδιού. Μετά την εκτέλεση της αναζήτησης, ο χρήστης ανακατευθύνεται σε σελίδα που εμφανίζει όλα τα παιχνίδια των οποίων ο τίτλος περιλαμβάνει το κείμενο που εισήγαγε.

Η αρχική σελίδα παρουσιάζει επίσης τις βασικές κατηγορίες των βιντεοπαιχνιδιών, επιτρέποντας στο χρήστη να μεταβεί σε σελίδα όπου εμφανίζονται όλες οι διαθέσιμες κατηγορίες και τα αντίστοιχα παιχνίδια.
Στη συνέχεια, η σελίδα εμφανίζει τρεις ξεχωριστές λίστες παιχνιδιών, κατηγοριοποιημένες ως εξής:

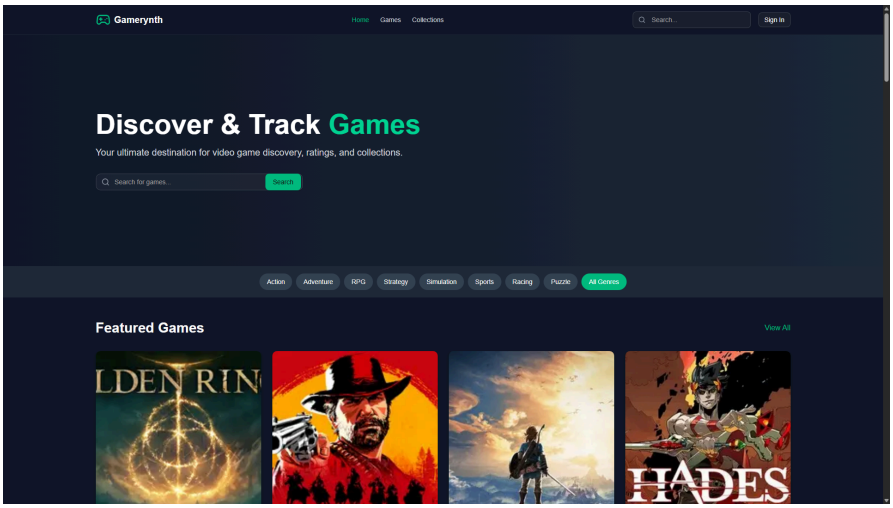
1. Featured Games: περιλαμβάνει τίτλους που επιλέχθηκαν με βάση τη συνδυαστική αξιολόγηση από τους χρήστες και την επισκεψιμότητα τους.

2. Recent Releases: εμφανίζει τα πιο πρόσφατα κυκλοφορήσαντα παιχνίδια, δίνοντας στον χρήστη τη δυνατότητα να ενημερωθεί για νέους τίτλους και να ανακαλύψει φρέσκο περιεχόμενο.

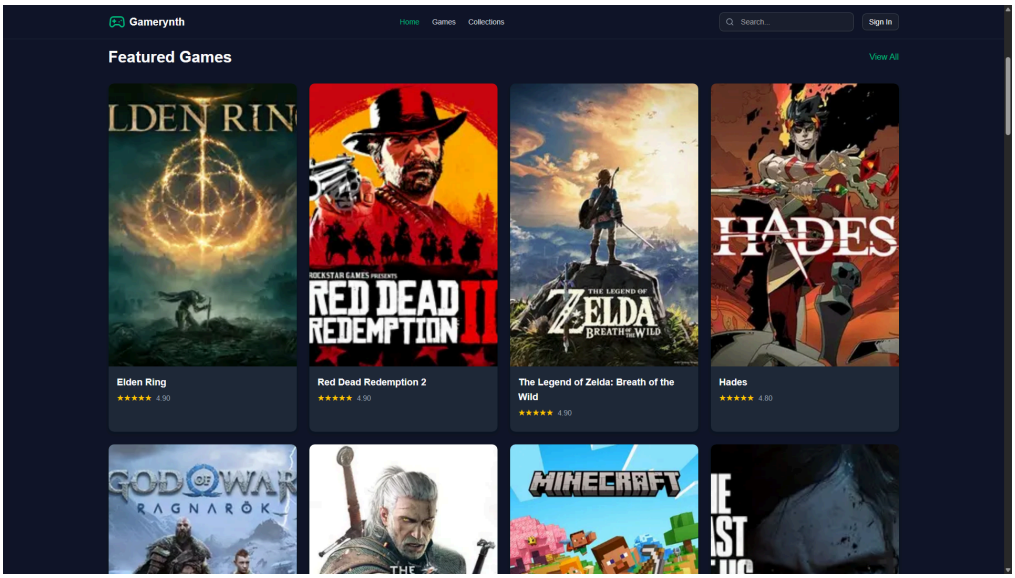
3. Popular Games: παρουσιάζει τα παιχνίδια με τη μεγαλύτερη δημοτικότητα, με βάση τον αριθμό αξιολογήσεων και τη μέση βαθμολογία τους, επιτρέποντας στους χρήστες να βλέπουν τους τίτλους που προτιμώνται από την κοινότητα.

Στο τέλος της σελίδας, εμφανίζονται τα δύο πιο δημοφιλή collections που έχουν δημιουργηθεί από χρήστες. Αυτά αποτελούν συνδυασμούς παιχνιδιών που έχουν συγκεντρώσει υψηλή αξιολόγηση ή έχουν προτιμηθεί από πολλούς χρήστες, προσφέροντας προτάσεις και ιδέες για νέα παιχνίδια που μπορεί να ενδιαφέρουν τον χρήστη.

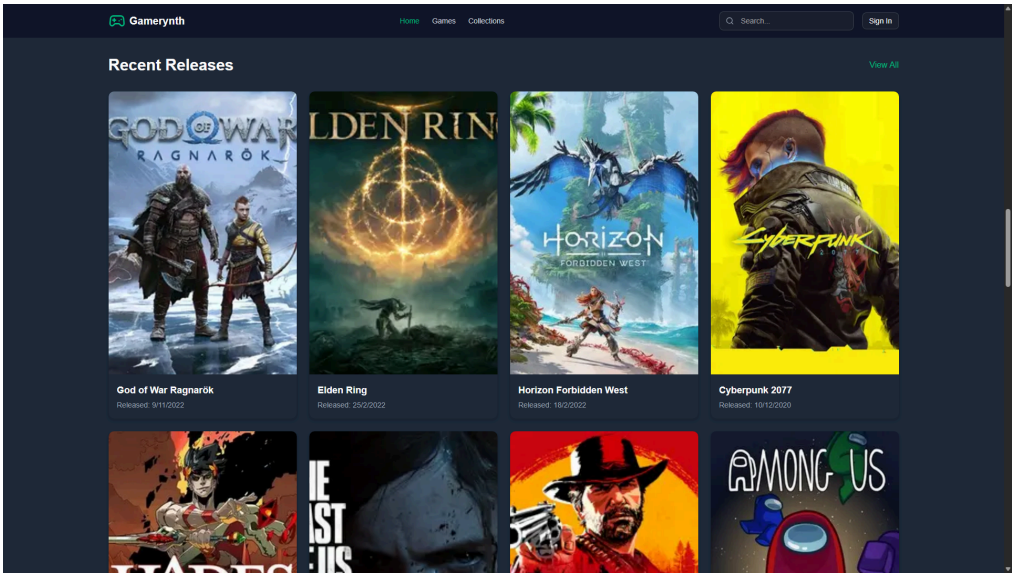
Το header της αρχικής σελίδας παρέχει δυνατότητες γρήγορης πλοήγησης, δίνοντας πρόσβαση στη διαδικασία σύνδεσης ή στην πλοήγηση σε άλλες βασικές σελίδες της πλατφόρμας, όπως η λίστα όλων των παιχνιδιών και η σελίδα των collections. Με αυτόν τον τρόπο διασφαλίζεται η εύκολη πρόσβαση σε όλες τις λειτουργίες και η βέλτιστη εμπειρία χρήσης.



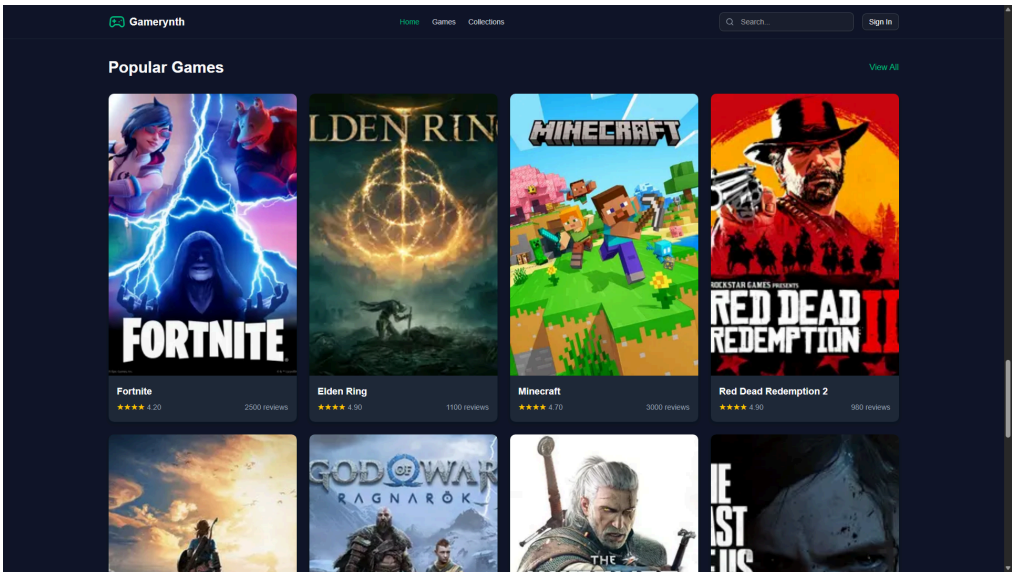
Εικόνα 4 Αρχική σελίδα εφαρμογής για επισκέπτη χωρίς να έχει γίνει αυθεντικοποίηση



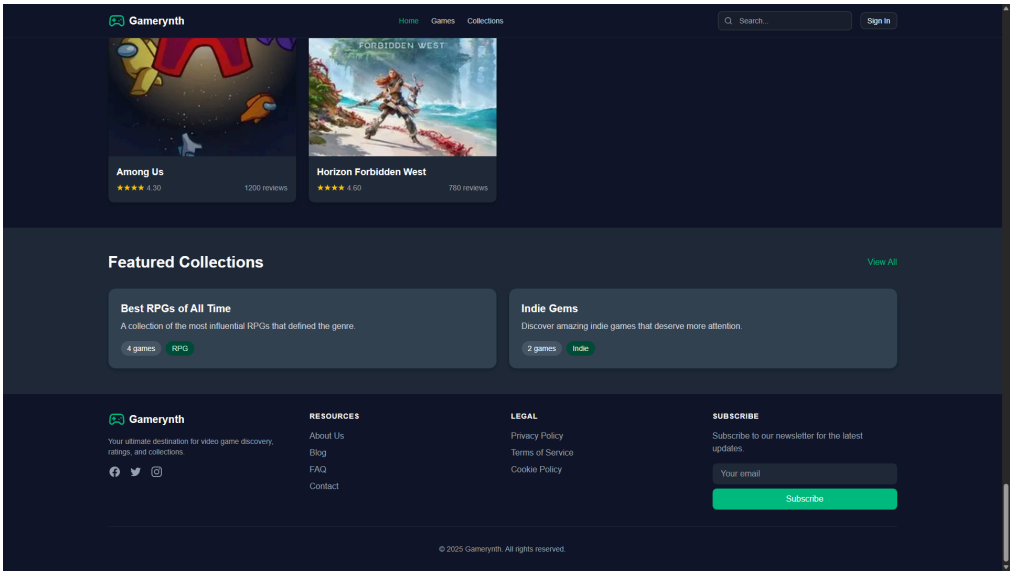
Εικόνα 5 Featured Games Section στην αρχική σελίδα



Εικόνα 6 Recent Releases Section στην αρχική σελίδα



Εικόνα 7 Popular Games Section στην αρχική σελίδα



Εικόνα 8 Footer εφαρμογής που εμφανίζεται σε όλες τις διαφορετικές σελίδες

3.3.3 Σελίδα κατηγοριών (Game Genres)

Η σελίδα των κατηγοριών (game genres) παρέχει στον χρήστη μια πλήρη επισκόπηση των διαθέσιμων ειδών παιχνιδιών και λειτουργεί ως κεντρικό σημείο πλοήγησης για την εξερεύνηση περιεχομένου ανά κατηγορία.

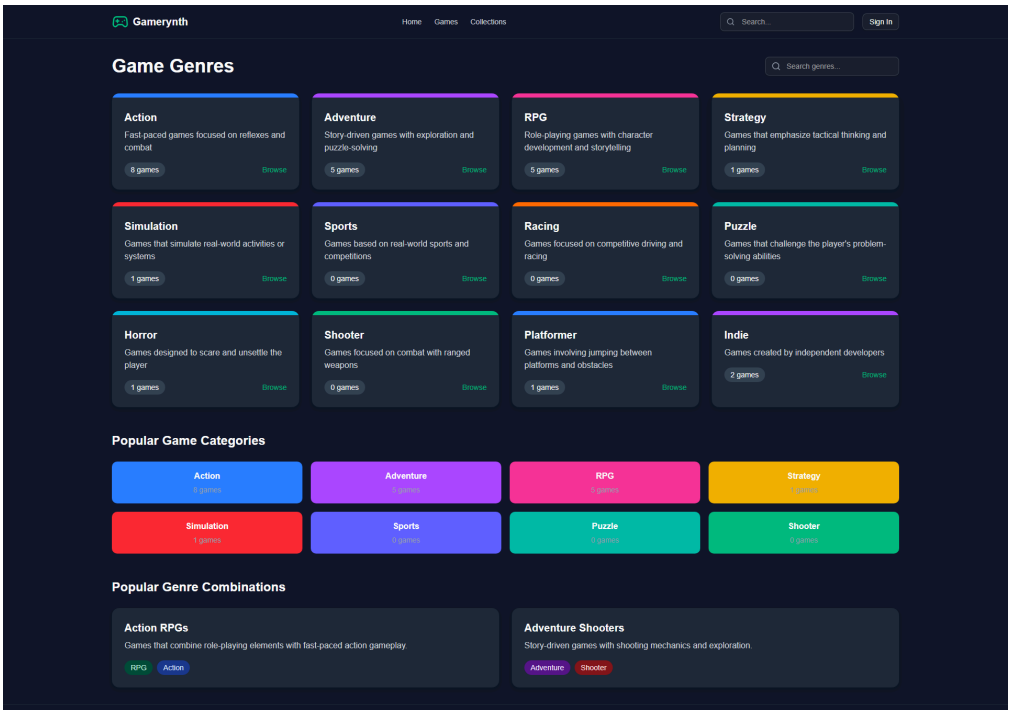
Στο πάνω μέρος της σελίδας παρουσιάζεται μια λίστα με όλες τις διαθέσιμες κατηγορίες, όπου για κάθε κατηγορία εμφανίζεται και ο αριθμός των παιχνιδιών που ανήκουν σε

αυτήν. Ο χρήστης μπορεί να επιλέξει οποιαδήποτε κατηγορία, με αποτέλεσμα να ανακατευθύνεται σε μια σελίδα που εμφανίζει τη λίστα των παιχνιδιών που ανήκουν αποκλειστικά σε αυτή την κατηγορία.

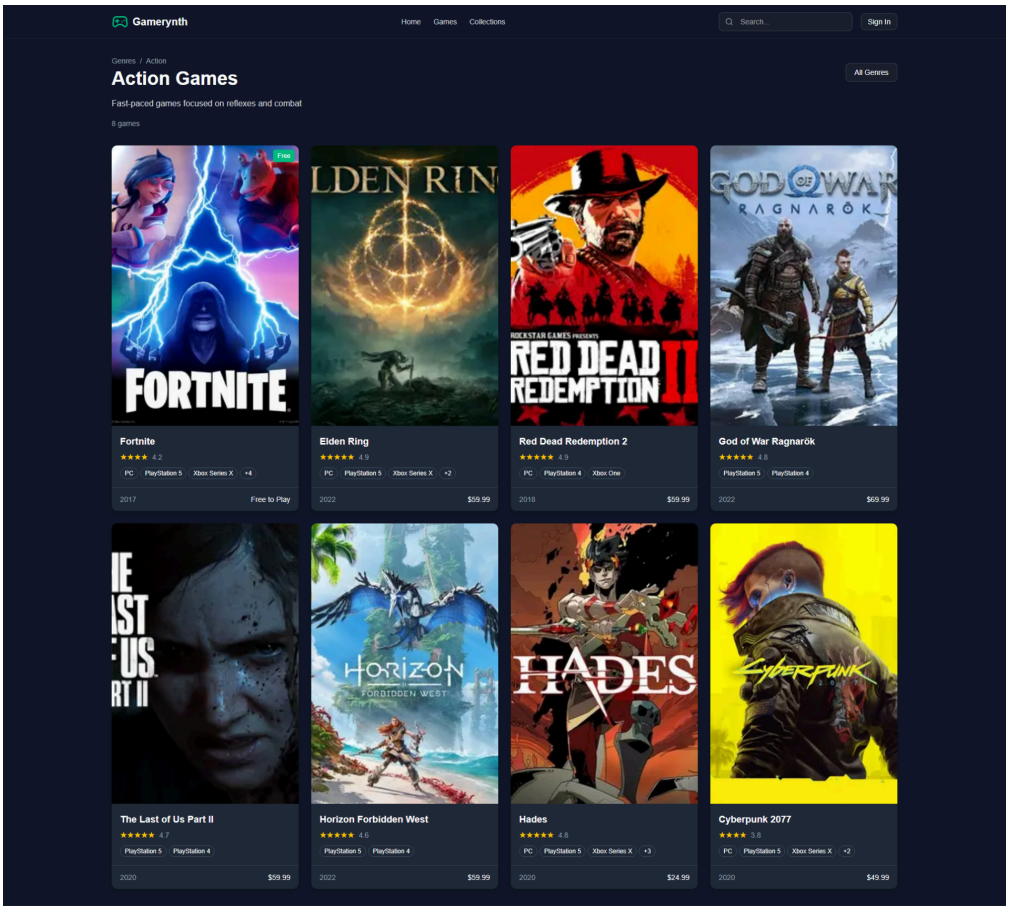
Ακολουθεί ένα δεύτερο section, το οποίο παρουσιάζει τα δημοφιλέστερα genres, βασισμένα στον αριθμό αξιολογήσεων και στη δημοτικότητα των παιχνιδιών που περιλαμβάνουν. Αυτό το τμήμα δίνει τη δυνατότητα στον χρήστη να εντοπίσει εύκολα τα είδη που προτιμώνται περισσότερο από την κοινότητα.

Στο τρίτο section παρουσιάζονται τα δημοφιλέστερα combinations κατηγοριών, δηλαδή συνδυασμοί δύο ή περισσότερων genres που εμφανίζονται συχνά στα παιχνίδια με υψηλή αξιολόγηση ή μεγάλη δημοτικότητα. Επιλέγοντας οποιοδήποτε από αυτά τα combinations, ο χρήστης ανακατευθύνεται σε μια λίστα με παιχνίδια που ανήκουν στον συγκεκριμένο συνδυασμό κατηγοριών, διευκολύνοντας την ανακάλυψη παιχνιδιών που ταιριάζουν σε πιο σύνθετες προτιμήσεις.

Η σελίδα κατηγοριών συμβάλλει στη διευκόλυνση της αναζήτησης και της οργάνωσης περιεχομένου, επιτρέποντας στους χρήστες να περιηγηθούν τόσο σε μεμονωμένα genres όσο και σε συνδυασμούς τους, ενώ παράλληλα προσφέρει στατιστικά στοιχεία για τη δημοτικότητα των κατηγοριών.



Εικόνα 9 Εσωτερική σελίδα κατηγοριών βιντεοπαιχνιδιών



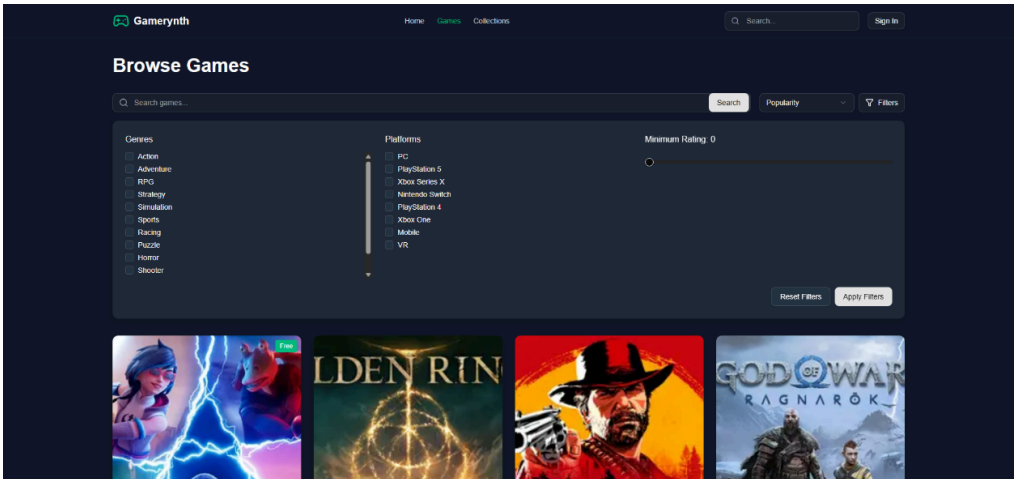
Εικόνα 10 Εσωτερική σελίδα συγκεκριμένης κατηγορίας βιντεοπαιχνιδιών

3.3.4 Σελίδα λίστας βιντεοπαιχνιδιών

Η σελίδα λίστας βιντεοπαιχνιδιών αποτελεί το κεντρικό σημείο αναζήτησης και περιήγησης για όλα τα παιχνίδια που είναι καταχωρημένα στην πλατφόρμα. Ο χρήστης μπορεί να δει τα παιχνίδια σε μορφή λίστας, με βασικές πληροφορίες για κάθε τίτλο, όπως όνομα, είδος (genre), πλατφόρμα, μέση βαθμολογία και χρόνο κυκλοφορίας.
Η σελίδα παρέχει δυνατότητες σύνθετης αναζήτησης και φιλτραρίσματος, ώστε ο χρήστης να βρίσκει πιο στοχευμένα αποτελέσματα:

- Φίλτρο ανά Genre: επιτρέπει την επιλογή ενός ή περισσότερων ειδών παιχνιδιών για περιορισμό της λίστας σε συγκεκριμένα είδη.
- Φίλτρο ανά Platform: επιτρέπει την επιλογή της επιθυμητής πλατφόρμας (π.χ. PC, PlayStation, Xbox) για εμφάνιση μόνο των αντίστοιχων παιχνιδιών.
- Φίλτρο Minimum Rating: δίνει τη δυνατότητα στον χρήστη να περιορίσει τη λίστα σε παιχνίδια που έχουν μέση αξιολόγηση ίση ή μεγαλύτερη από την επιλεγμένη τιμή.
- Πεδία αναζήτησης τίτλου: ο χρήστης μπορεί να πληκτρολογήσει τον τίτλο ή μέρος του τίτλου ενός παιχνιδιού για άμεση εύρεση συγκεκριμένων παιχνιδιών.

Η συνδυαστική χρήση των φίλτρων και του πεδίου αναζήτησης επιτρέπει στον χρήστη να εντοπίσει με ακρίβεια τα παιχνίδια που τον ενδιαφέρουν, ακόμη και όταν η βάση δεδομένων περιλαμβάνει μεγάλο αριθμό τίτλων. Η σελίδα ενημερώνει δυναμικά τη λίστα αποτελεσμάτων καθώς ο χρήστης εφαρμόζει τα φίλτρα ή πληκτρολογεί κείμενο στο πεδίο αναζήτησης, παρέχοντας άμεση και διαδραστική εμπειρία χρήσης.



Εικόνα 11 Φίλτρα αναζήτησης στις λίστες βιντεοπαιχνιδιών

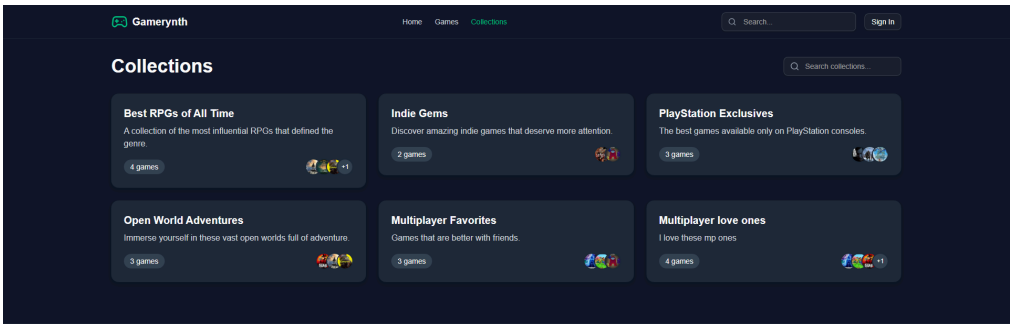
3.3.5 Σελίδα Collections

Η σελίδα Collections παρέχει στον χρήστη δυνατότητα προβολής και διαχείρισης συλλογών βιντεοπαιχνιδιών, τόσο των δικών του όσο και αυτών που έχουν δημιουργήσει άλλοι χρήστες. Κάθε collection αποτελεί μια ομάδα παιχνιδιών που έχει οργανωθεί γύρω από συγκεκριμένο θέμα ή προτίμηση, όπως είδος, δημοτικότητα ή προσωπικό ενδιαφέρον.

Η σελίδα εμφανίζει τις συλλογές σε μορφή λίστας, με βασικές πληροφορίες για κάθε collection, όπως τίτλος, περιγραφή, δημιουργός και αριθμός παιχνιδιών που περιλαμβάνει. Ο χρήστης μπορεί να επιλέξει οποιοδήποτε collection για να δει τη λεπτομερή λίστα των παιχνιδιών που περιέχει.

Η αναζήτηση στη σελίδα collections επιτρέπει στον χρήστη να εντοπίσει collections με βάση το όνομα. Η λίστα ενημερώνεται δυναμικά καθώς εφαρμόζονται τα κριτήρια αναζήτησης, διευκολύνοντας την εύρεση συγκεκριμένων συλλογών σε μεγάλες βάσεις δεδομένων.

Η σελίδα Collections υποστηρίζει επίσης την αλληλεπίδραση με τις συλλογές του ίδιου του χρήστη, επιτρέποντας την προβολή, την επεξεργασία και τη διαγραφή των δικών του collections, ενισχύοντας έτσι τη διαδραστική και εξατομικευμένη εμπειρία χρήσης της πλατφόρμας.



Εικόνα 12 Σελίδα με τις λίστες βιντεοπαιχνιδιών χρηστών

3.3.6 Εσωτερική Σελίδα Collection

Η εσωτερική σελίδα ενός collection παρέχει λεπτομερή πληροφορία για τη συγκεκριμένη συλλογή και αποτελεί σημείο αναφοράς για τον χρήστη που επιθυμεί να εξερευνήσει τα παιχνίδια που περιλαμβάνει. Στην κορυφή της σελίδας εμφανίζονται τα βασικά στοιχεία του collection:

- Τίτλος: το όνομα της συλλογής.
- Περιγραφή: σύντομη περιγραφή του σκοπού ή του θέματος της συλλογής.
- Δημιουργός και ημερομηνία δημιουργίας: πληροφορίες σχετικά με τον χρήστη που δημιούργησε το collection και την ημερομηνία που έγινε η δημιουργία.
- Σύνολο παιχνιδιών: ο αριθμός των τίτλων που περιλαμβάνει η συλλογή.

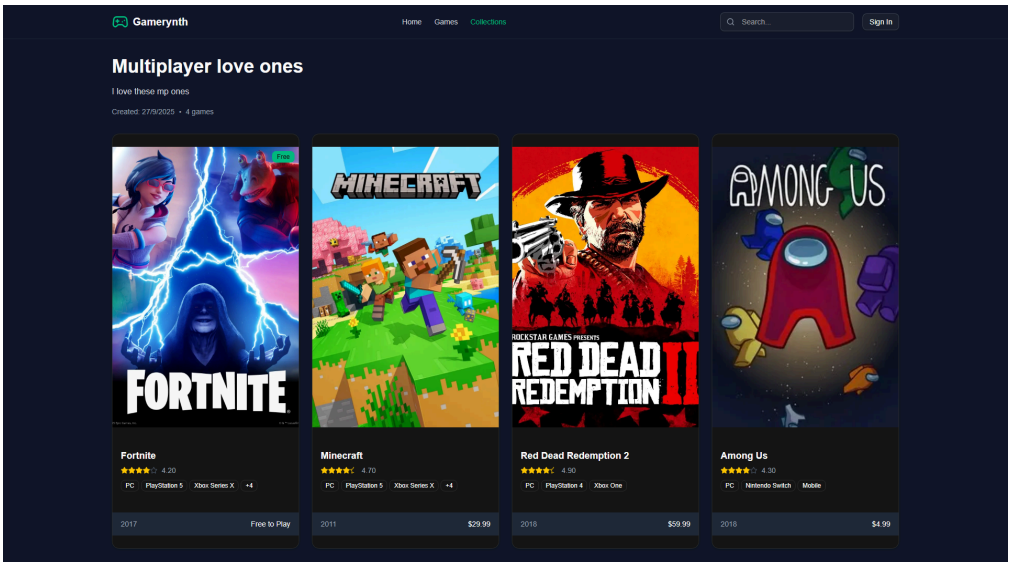
Στη συνέχεια, εμφανίζεται η λίστα με τα παιχνίδια που ανήκουν στο collection, παρέχοντας βασικές πληροφορίες για κάθε παιχνίδι, όπως είδος, πλατφόρμα και μέση βαθμολογία. Ο χρήστης μπορεί να επιλέξει οποιοδήποτε παιχνίδι για να μεταβεί στη σελίδα λεπτομερειών του.
Για τους συνδεδεμένους χρήστες, η σελίδα προσφέρει δύο επιπλέον λίστες προτάσεων βασισμένων σε Συστήματα Προτάσεων (Recommendation Systems), ώστε να διευκολυνθεί η ανακάλυψη νέων παιχνιδιών:

1. Content-based filtering: εμφανίζονται προτάσεις παιχνιδιών που βασίζονται στα χαρακτηριστικά των τίτλων που έχει ήδη αξιολογήσει ή προσθέσει ο χρήστης στο collection, όπως είδος παιχνιδιού, tags ή game modes.
2. Collaborative filtering: εμφανίζονται προτάσεις βασισμένες σε συγκρίσεις προτιμήσεων με άλλους χρήστες ή collections με παρόμοιο προφίλ, επιτρέποντας την ανακάλυψη τίτλων που προτείνονται από την κοινότητα.

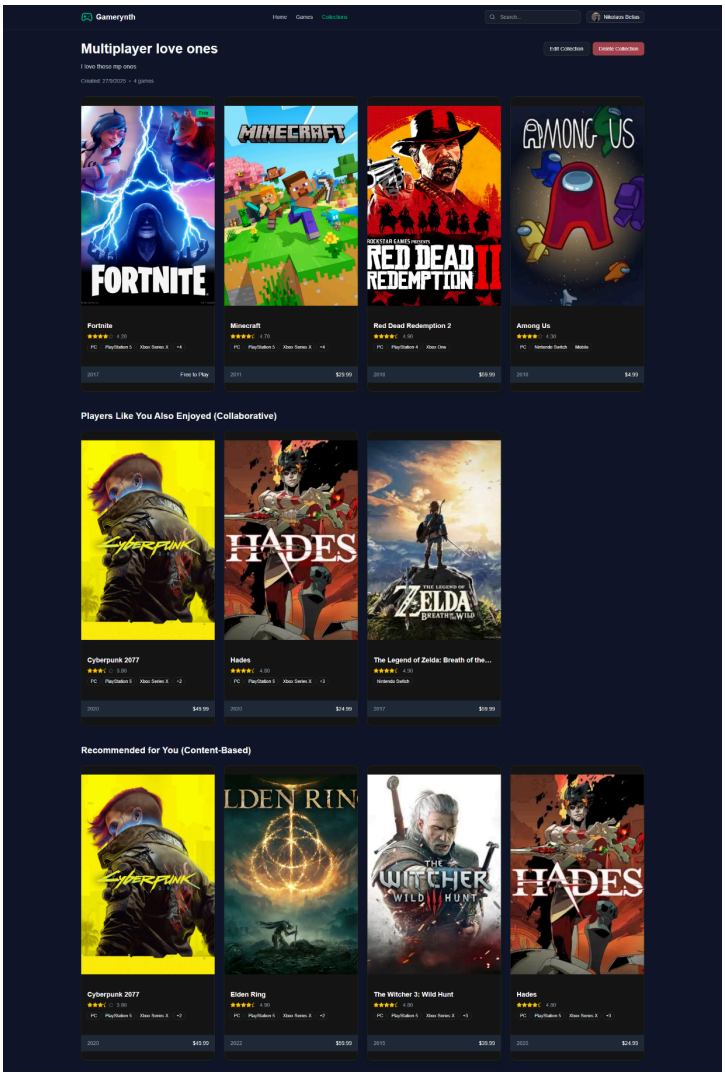
Επιπλέον, ο χρήστης που είναι δημιουργός της συλλογής διαθέτει λειτουργίες επεξεργασίας, όπως:

- Αφαίρεση παιχνιδιών από τη συλλογή.
- Διαγραφή ολόκληρου του collection.

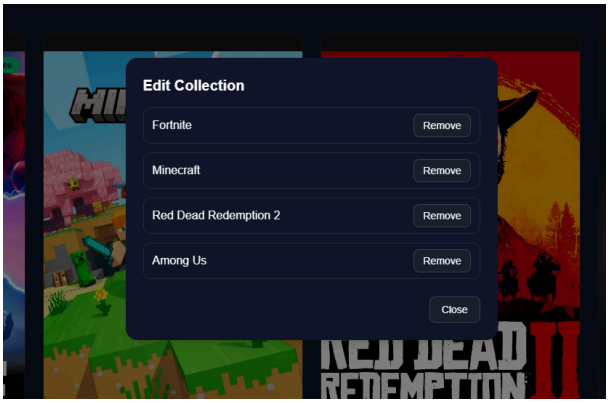
Με τον τρόπο αυτό, η εσωτερική σελίδα του collection συνδυάζει την παρουσίαση δομημένων πληροφοριών για τη συλλογή με μηχανισμούς εξατομικευμένης ανακάλυψης περιεχομένου και δυνατότητες διαχείρισης, προσφέροντας μια ολοκληρωμένη και διαδραστική εμπειρία χρήσης.



Εικόνα 13 Εσωτερική σελίδα λίστας βιντεοπαιχνιδιών επισκέπτη



Εικόνα 14 Εσωτερική σελίδα λίστας βιντεοπαιχνιδιών συνδεδεμένου χρήστη



Εικόνα 15 Επεξεργασία βιντεοπαιχνιδιών λίστας

3.3.7 Εσωτερική Σελίδα Βιντεοπαιχνιδιού

Η εσωτερική σελίδα ενός βιντεοπαιχνιδιού παρέχει στον χρήστη όλες τις διαθέσιμες πληροφορίες σχετικά με τον τίτλο, οργανωμένες με τρόπο που διευκολύνει την ανάγνωση και την αλληλεπίδραση. Η σελίδα περιλαμβάνει τα εξής τμήματα:

1. Βασικές πληροφορίες
   - Τίτλος και περιγραφή του παιχνιδιού.
   - Ημερομηνία κυκλοφορίας (release_date).
   - Genres και πλατφόρμες στις οποίες διατίθεται το παιχνίδι.
   - Developer και Publisher.
   - Μέση βαθμολογία (average_rating), αριθμός αξιολογήσεων χρηστών (user_reviews_count) και αριθμός κριτικών από ειδικούς (critic_reviews_count).
   - Δείκτης δημοφιλίας (popularity_score) για γρήγορη εκτίμηση της αποδοχής του τίτλου.
   - Ηλικιακή κατάταξη (age_rating), τιμή αγοράς (price) και αν το παιχνίδι είναι free-to-play.
   - DLCs που έχουν κυκλοφορήσει και υποστήριξη multiplayer / game modes.
   - Cover image, σύνδεσμος για trailer και επίσημο website.

2. Κριτικές χρηστών

Οι συνδεδεμένοι χρήστες έχουν τη δυνατότητα να προσθέτουν κριτικές για το παιχνίδι. Κάθε κριτική περιλαμβάνει:

- Περιγραφή / σχόλιο.
- Βαθμολογία σε κλίμακα 1 έως 5 αστεριών.

Οι κριτικές εμφανίζονται κάτω από τις βασικές πληροφορίες, επιτρέποντας στους χρήστες να δουν τη συλλογική γνώμη της κοινότητας.

3. Media

Η σελίδα παρέχει οπτικό υλικό για καλύτερη εμπειρία χρήσης:

- Screenshots του παιχνιδιού, που εμφανίζονται σε γκαλερί.
- Trailer μέσω ανακατεύθυνσης χρήστη.

4. DLCs

Ο χρήστης μπορεί να δει τη λίστα των DLCs του παιχνιδιού, με βασικές πληροφορίες όπως όνομα, κόστος και περιγραφή.

5. Διαχείριση Collections

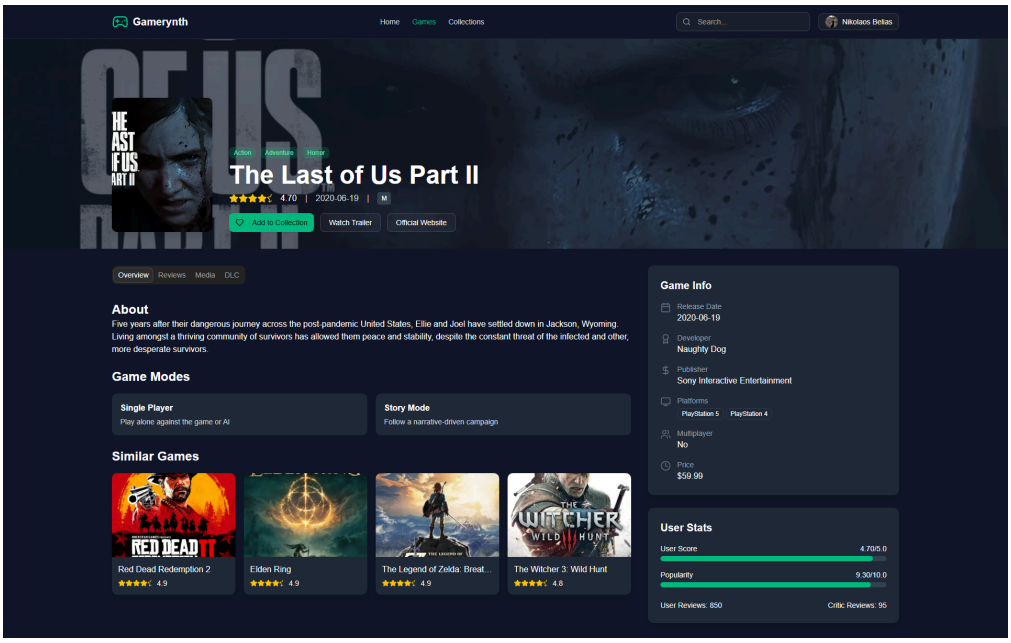
Η σελίδα επιτρέπει την διαχείριση του παιχνιδιού σε collections:

- Προσθήκη του παιχνιδιού σε υπάρχουσα συλλογή.
- Αφαίρεση του παιχνιδιού από μια συλλογή.
- Δημιουργία νέας συλλογής και ταυτόχρονη προσθήκη του παιχνιδιού σε αυτή.

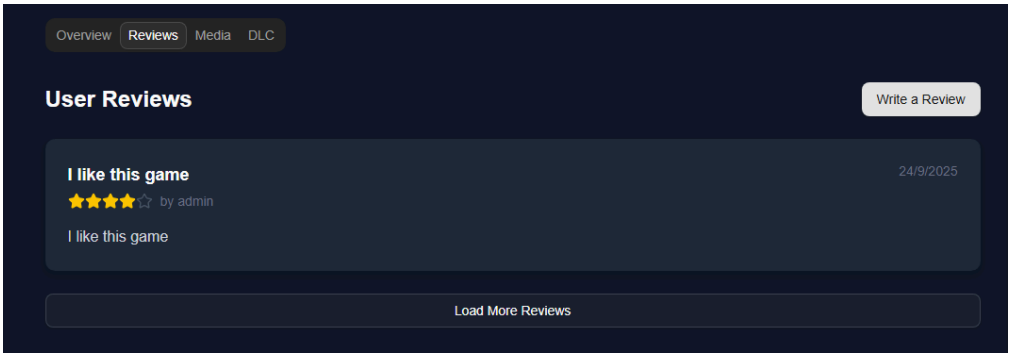
Αυτές οι λειτουργίες ενισχύουν τη δυνατότητα εξατομίκευσης της εμπειρίας χρήστη, επιτρέποντας την οργάνωση και αποθήκευση των παιχνιδιών με βάση τα προσωπικά ενδιαφέροντα.

6. Εμφάνιση παρόμοιων τίτλων

Στους χρήστες μφανίζονται επίσης προτάσεις που μοιάζουν με το τρέχον, βάσει χαρακτηριστικών όπως είδος, tags ή game modes.

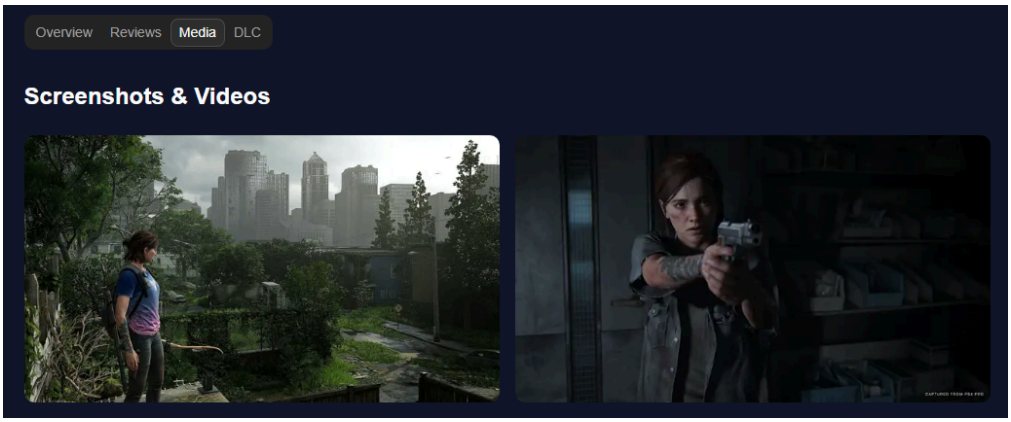


Εικόνα 16 Εσωτερική σελίδα πληροφοριών βιντεοπαιχνιδιού

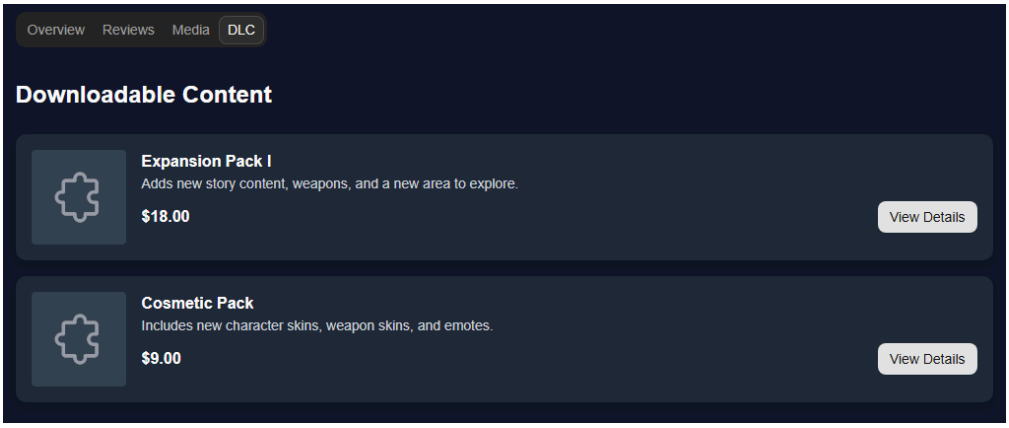


Εικόνα 17 Αξιολογήσεις χρηστών

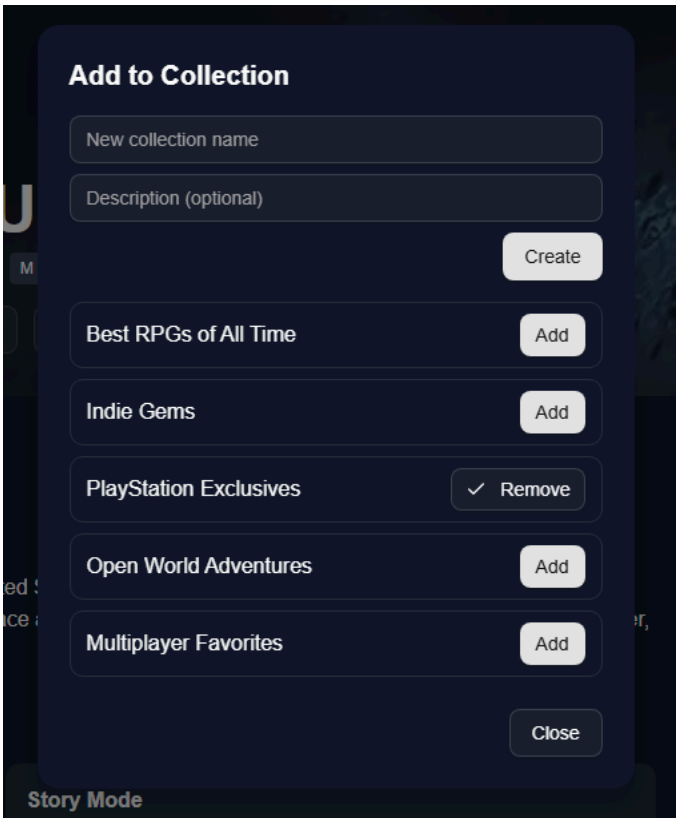
Εικόνα 18 Φόρμα αξιολόγησης βιντεοπαιχνιδιού



Εικόνα 19 Media βιντεοπαιχνιδιού



Εικόνα 20 Λίστα με διαθέσιμα DLC βιντεοπαιχνιδιού



Εικόνα 21 Προσθήκη / αφαίρεση βιντεοπαιχνιδιού σε λίστα και δημιουργία νέας λίστας

3.4 Διαδικασία εγκατάστασης και τοπικής εκτέλεσης

Στην παρούσα ενότητα περιγράφεται αναλυτικά η διαδικασία εγκατάστασης και τοπικής εκτέλεσης της διαδικτυακής πλατφόρμας. Στόχος είναι να δοθούν όλες οι απαραίτητες οδηγίες ώστε ένας προγραμματιστής ή αξιολογητής να μπορεί να εκτελέσει την εφαρμογή στο τοπικό του περιβάλλον, χωρίς να απαιτείται χειροκίνητη εγκατάσταση των επιμέρους τεχνολογιών.

Για την υλοποίηση και εκτέλεση της πλατφόρμας χρησιμοποιείται η τεχνολογία Docker, η οποία επιτρέπει την απομόνωση των επιμέρους υπηρεσιών (frontend, backend, βάση δεδομένων) σε ανεξάρτητα containers και διασφαλίζει ενιαίο και αναπαραγώγιμο περιβάλλον εκτέλεσης.

3.4.1 Εισαγωγή στο Docker και το Docker Compose

Το Docker είναι μια πλατφόρμα εικονικοποίησης σε επίπεδο εφαρμογής (containerization), η οποία επιτρέπει τη δημιουργία ελαφρών και απομονωμένων περιβαλλόντων εκτέλεσης. Κάθε υπηρεσία της εφαρμογής εκτελείται μέσα σε ένα container, το οποίο περιλαμβάνει όλες τις απαραίτητες εξαρτήσεις.

Το Docker Compose χρησιμοποιείται για τον συντονισμό πολλαπλών containers και επιτρέπει την εκκίνηση ολόκληρης της πλατφόρμας με μία μόνο εντολή. Στην παρούσα εφαρμογή, το Docker Compose χρησιμοποιείται για:

- το frontend (Next.js),
- το backend (Django Rest Framework),
- τη βάση δεδομένων (PostgreSQL).

3.4.2 Προαπαιτούμενα

Για την τοπική εκτέλεση της εφαρμογής απαιτείται:

- Εγκατάσταση του Docker Desktop (διαθέσιμο για Windows, macOS και Linux).
- Ενεργοποίηση του Docker Daemon (μέσω Docker Desktop).
- Πρόσβαση στον πηγαίο κώδικα της εφαρμογής.

Δεν απαιτείται τοπική εγκατάσταση Node.js, Python ή PostgreSQL, καθώς όλες οι εξαρτήσεις διαχειρίζονται από τα Docker containers.

3.4.3 Ρύθμιση Περιβάλλοντος Frontend

Για το frontend της εφαρμογής (Next.js) απαιτείται η δημιουργία ενός αρχείου περιβάλλοντος .env στο root directory του frontend project.

Το αρχείο .env περιλαμβάνει τις παρακάτω μεταβλητές:

```
GOOGLE_CLIENT_ID=***.apps.googleusercontent.com

GOOGLE_CLIENT_SECRET=***

NEXTAUTH_SECRET=WERn24njGcj234C2n24nx3rWERCGTj

NEXT_PUBLIC_BACKEND_URL=http://localhost:8001
```

Οι μεταβλητές αυτές χρησιμοποιούνται για:

- την υλοποίηση του Google Authentication μέσω NextAuth,
- την ασφάλεια των sessions χρηστών,
- την επικοινωνία του frontend με το backend REST API.

Το αρχείο .env διαβάζεται αυτόματα από το Docker container του frontend κατά την εκκίνηση της εφαρμογής.

3.4.4 Ρύθμιση Περιβάλλοντος Backend

Για το backend της εφαρμογής (Django Rest Framework) απαιτείται η δημιουργία δύο αρχείων περιβάλλοντος στον φάκελο .env.

Αρχείο .env_app

Το αρχείο .env_app περιλαμβάνει ρυθμίσεις που αφορούν τη λειτουργία του Django:

```
DJANGO_SETTINGS_MODULE=config.settings.local

SECRET_KEY=gjiogjiao+dfnocnfiaouieoncifoadofnufiao^s

DEBUG=1

DJANGO_ALLOWED_HOSTS=localhost

DJANGO_ADMIN_URL=admin

DJANGO_ADMIN_USERNAME=admin

DJANGO_ADMIN_PASSWORD=admin

DJANGO_ADMIN_EMAIL=admin@example.com

DJANGO_SECURE_SSL_REDIRECT=False
```

Οι παραπάνω μεταβλητές καθορίζουν:
το περιβάλλον ρυθμίσεων του Django,
την ασφάλεια της εφαρμογής,
τις ρυθμίσεις debugging,
τη δημιουργία λογαριασμού διαχειριστή (admin).

Αρχείο .env_db

Το αρχείο .env_db περιλαμβάνει τις ρυθμίσεις της βάσης δεδομένων PostgreSQL:

```
POSTGRES_USER=ac_db_dev

POSTGRES_PASSWORD=ac_db_dev

POSTGRES_DB=ac_db_dev
```

Οι μεταβλητές αυτές χρησιμοποιούνται τόσο από το container της βάσης δεδομένων όσο και από το backend container για την επικοινωνία με αυτή.

3.4.5 Εκκίνηση της Εφαρμογής με Docker Compose

Αφού έχουν δημιουργηθεί όλα τα απαραίτητα αρχεία περιβάλλοντος, η εφαρμογή μπορεί να εκτελεστεί τοπικά με την ακόλουθη εντολή:

```
docker compose -f docker-compose-local.yml up --build
```

Η εντολή αυτή:

- δημιουργεί τα Docker images για frontend, backend και database,
- εκκινεί όλα τα απαραίτητα containers,
- συνδέει τις υπηρεσίες μέσω εσωτερικού Docker δικτύου,
- εκτελεί migrations και αρχικοποιήσεις όπου απαιτείται.

Μετά την επιτυχή εκτέλεση της εντολής:

- το frontend είναι διαθέσιμο στον browser στη διεύθυνση http://localhost:3000,
- το backend API είναι διαθέσιμο στη διεύθυνση http://localhost:8001,
- η βάση δεδομένων λειτουργεί σε απομονωμένο container χωρίς άμεση έκθεση στον χρήστη.

3.4.6 Πλεονεκτήματα της Προσέγγισης

Η χρήση Docker και Docker Compose προσφέρει:

- ενιαίο περιβάλλον εκτέλεσης ανεξαρτήτως λειτουργικού συστήματος,
- εύκολη εγκατάσταση και εκκίνηση της εφαρμογής,
- απομόνωση υπηρεσιών και αυξημένη ασφάλεια,
- δυνατότητα μελλοντικής ανάπτυξης και μετάβασης σε περιβάλλον παραγωγής.

Η συγκεκριμένη προσέγγιση καθιστά την πλατφόρμα εύκολα επεκτάσιμη και συντηρήσιμη, ενώ μειώνει σημαντικά τα προβλήματα που σχετίζονται με τη διαχείριση εξαρτήσεων.

3.5 Τεχνολογικό περιβάλλον

Η υλοποίηση της πλατφόρμας βασίστηκε στις ακόλουθες τεχνολογίες:

- Next.js και React για το frontend.
- TypeScript για αυξημένη αξιοπιστία κώδικα.

- Django Rest Framework για την υλοποίηση του backend API.
- PostgreSQL για την αποθήκευση των δεδομένων.
- Εξωτερικό API (IGDB) για τον αρχικό εμπλουτισμό δεδομένων.

3.6 Σύνοψη κεφαλαίου

Στο κεφάλαιο αυτό παρουσιάστηκε το εγχειρίδιο χρήσης και εγκατάστασης της διαδικτυακής πλατφόρμας. Περιγράφηκαν οι βασικές λειτουργίες του συστήματος, οι ρόλοι χρηστών, η ροή χρήσης της εφαρμογής και τα απαραίτητα βήματα για την τοπική εκτέλεσή της. Το κεφάλαιο αυτό παρέχει στον αναγνώστη μια ολοκληρωμένη εικόνα του τελικού αποτελέσματος της υλοποίησης, λειτουργώντας ως γέφυρα προς τα επόμενα κεφάλαια της εργασίας.

Κεφάλαιο 4ο

Αρχιτεκτονική και υλοποίηση της Πλατφόρμας

4.1 Εισαγωγή

Στο παρόν κεφάλαιο παρουσιάζεται αναλυτικά η αρχιτεκτονική της πλατφόρμας καθώς και η διαδικασία υλοποίησής της από το αρχικό στάδιο σχεδιασμού έως την τελική ολοκλήρωση. Ιδιαίτερη έμφαση δίνεται στη σταδιακή ανάπτυξη του συστήματος, στις τεχνολογικές επιλογές που πραγματοποιήθηκαν και στη λογική πίσω από κάθε βήμα, ώστε να αποτυπωθεί με σαφήνεια η συνολική πορεία υλοποίησης.

4.2 Γενική Αρχιτεκτονική Συστήματος

Η εφαρμογή υλοποιήθηκε με αρχιτεκτονική client–server, ακολουθώντας τη φιλοσοφία του διαχωρισμού αρμοδιοτήτων (separation of concerns). Το σύστημα αποτελείται από τρία βασικά υποσυστήματα:

1. Frontend εφαρμογή, υπεύθυνη για την αλληλεπίδραση με τον χρήστη.
2. Backend εφαρμογή, η οποία υλοποιεί την επιχειρησιακή λογική και παρέχει δεδομένα μέσω REST API.
3. Βάση δεδομένων, για τη μόνιμη αποθήκευση των δεδομένων.

Η επικοινωνία μεταξύ frontend και backend πραγματοποιείται αποκλειστικά μέσω HTTP requests, ενώ η αυθεντικοποίηση χρηστών βασίζεται σε σύγχρονες μεθόδους OAuth.

4.3 Αναλυτική περιγραφή της Διαδικασίας Υλοποίησης

4.3.1 Καθορισμός Στόχων και Ανάλυση Απαιτήσεων

Η υλοποίηση ξεκίνησε με τον καθορισμό του βασικού στόχου της εφαρμογής: τη δημιουργία μιας web πλατφόρμας όπου οι χρήστες μπορούν να ανακαλύπτουν, να αξιολογούν και να οργανώνουν βιντεοπαιχνίδια μέσω συλλογών (collections).

Στο στάδιο αυτό καταγράφηκαν οι κύριες λειτουργικές απαιτήσεις, όπως:

- προβολή και αναζήτηση βιντεοπαιχνιδιών,
- φιλτράρισμα με βάση είδος, πλατφόρμα και βαθμολογία,
- δημιουργία και διαχείριση collections,
- σύστημα αξιολογήσεων και κριτικών,
- εξατομικευμένες προτάσεις παιχνιδιών.

Παράλληλα, ορίστηκαν και μη λειτουργικές απαιτήσεις, όπως:

- επεκτασιμότητα,
- ευχρηστία,
- ασφάλεια,
- δυνατότητα εύκολης τοπικής εκτέλεσης.

4.3.2 Επιλογή Τεχνολογιών

Με βάση τις παραπάνω απαιτήσεις επιλέχθηκαν τεχνολογίες που προσφέρουν σταθερότητα και ευελιξία:

- Next.js για το frontend, λόγω της υποστήριξης server-side rendering και της καλής απόδοσης.
- Django & Django Rest Framework για το backend, χάρη στη δομημένη αρχιτεκτονική και τις έτοιμες λύσεις που παρέχει.
- PostgreSQL ως σχεσιακή βάση δεδομένων.
- Docker & Docker Compose για την ομοιομορφία του περιβάλλοντος ανάπτυξης.
- Google Authentication για ασφαλή και απλοποιημένη αυθεντικοποίηση.

4.3.3 Σταδιακή Ανάπτυξη Backend

Η ανάπτυξη ξεκίνησε από το backend, καθώς αποτελεί τον βασικό πυρήνα της εφαρμογής.

Αρχικά δημιουργήθηκε το βασικό Django project και οργανώθηκε σε επιμέρους εφαρμογές, ώστε κάθε λειτουργικό κομμάτι να είναι ανεξάρτητο. Στη συνέχεια σχεδιάστηκε το μοντέλο δεδομένων, δίνοντας έμφαση:

- στις σωστές σχέσεις μεταξύ οντοτήτων,
- στη μελλοντική επεκτασιμότητα,
- στην αποδοτική εκτέλεση ερωτημάτων.

Η υλοποίηση των models αποτέλεσε τη βάση πάνω στην οποία χτίστηκε ολόκληρη η εφαρμογή.

4.3.4 Δημιουργία και Εξέλιξη REST API

Αφού ολοκληρώθηκε το βασικό data model, ξεκίνησε η υλοποίηση του REST API.

Σε αυτό το στάδιο:

- δημιουργήθηκαν serializers για κάθε οντότητα,
- ορίστηκαν endpoints για ανάκτηση, δημιουργία και τροποποίηση δεδομένων,
- προστέθηκαν φίλτρα και μηχανισμοί αναζήτησης,

- εφαρμόστηκε pagination για τη διαχείριση μεγάλου όγκου δεδομένων.

Η ανάπτυξη του API έγινε σταδιακά, με συνεχή δοκιμή των endpoints ώστε να διασφαλιστεί η σωστή λειτουργία τους.

4.3.5 Υλοποίηση Αυθεντικοποίησης και Ρόλων Χρηστών

Σε επόμενο στάδιο προστέθηκε το σύστημα αυθεντικοποίησης χρηστών. Επιλέχθηκε η χρήση Google Authentication, ώστε να απλοποιηθεί η διαδικασία εισόδου και να αυξηθεί το επίπεδο ασφάλειας.

Το backend ρυθμίστηκε ώστε:

- να αναγνωρίζει τον συνδεδεμένο χρήστη,
- να ελέγχει δικαιώματα πρόσβασης,
- να επιτρέπει συγκεκριμένες ενέργειες μόνο σε αυθεντικοποιημένους χρήστες.

4.3.6 Ανάπτυξη Frontend και Διασύνδεση με το Backend

Αφού το backend απέκτησε βασική λειτουργικότητα, ξεκίνησε η ανάπτυξη του frontend.

Η υλοποίηση πραγματοποιήθηκε σταδιακά:

- αρχικά δημιουργήθηκε το βασικό layout της εφαρμογής,
- στη συνέχεια υλοποιήθηκαν οι βασικές σελίδες,
- προστέθηκαν οι σελίδες λιστών και εσωτερικών προβολών.

Ιδιαίτερη προσοχή δόθηκε στη σωστή κατανάλωση του API και στη διαχείριση της κατάστασης της εφαρμογής.

4.3.7 Υλοποίηση Προηγμένων Λειτουργιών

Σε επόμενο στάδιο προστέθηκαν πιο σύνθετες λειτουργίες:

- σύστημα αξιολογήσεων,
- διαχείριση collections,
- συστήματα προτάσεων (content-based και collaborative filtering),
- δυναμική εμφάνιση DLCs και screenshots.

Οι λειτουργίες αυτές ενσωματώθηκαν σταδιακά, ώστε να ελέγχεται η ορθότητα και η απόδοσή τους.

4.3.8 Τελική Ολοκλήρωση και Containerization

Στο τελικό στάδιο πραγματοποιήθηκε η πλήρης ενοποίηση όλων των υποσυστημάτων. Παράλληλα, η εφαρμογή μεταφέρθηκε σε περιβάλλον Docker.

Η χρήση Docker επέτρεψε:

- την εύκολη εγκατάσταση του συστήματος,
- την αναπαραγωγιμότητα του περιβάλλοντος ανάπτυξης,
- την απλοποίηση της τοπικής εκτέλεσης.

Με τη χρήση Docker Compose, ολόκληρη η εφαρμογή μπορεί να ξεκινήσει με μία εντολή, γεγονός που ολοκληρώνει τη διαδικασία υλοποίησης.

4.3.9 Σχεδιαστικές Αποφάσεις και Τεχνικές Επιλογές

Κατά τη διάρκεια της υλοποίησης λήφθηκαν συγκεκριμένες σχεδιαστικές αποφάσεις με στόχο την απλότητα, την επεκτασιμότητα και τη συντηρησιμότητα του συστήματος.

Η επιλογή αρχιτεκτονικής client–server κρίθηκε απαραίτητη ώστε το frontend και το backend να εξελίσσονται ανεξάρτητα, επιτρέποντας μελλοντική αντικατάσταση ή επέκταση επιμέρους υποσυστημάτων.

Παράλληλα, η χρήση REST API διευκολύνει την πιθανή μελλοντική ανάπτυξη εναλλακτικών clients, όπως mobile εφαρμογών, χωρίς αλλαγές στον πυρήνα του συστήματος.

4.4 Συνολική Αποτίμηση

Η υλοποίηση της πλατφόρμας ακολούθησε μια δομημένη και εξελικτική προσέγγιση, επιτρέποντας τη συνεχή βελτίωση και επέκταση του συστήματος. Η επιλογή σύγχρονων τεχνολογιών και η σταδιακή ανάπτυξη συνέβαλαν στη δημιουργία μιας ολοκληρωμένης και επεκτάσιμης web εφαρμογής.

Κεφάλαιο 5ο

Συμπεράσματα

5.1 Σύνοψη Εργασίας

Στόχος της παρούσας διπλωματικής εργασίας ήταν η δημιουργία μιας διαδικτυακής πλατφόρμας για την ανακάλυψη, αξιολόγηση και οργάνωση βιντεοπαιχνιδιών μέσω collections, με ενσωμάτωση εξατομικευμένων συστημάτων προτάσεων. Η πλατφόρμα επιτρέπει στους χρήστες να αναζητούν παιχνίδια με βάση τον τίτλο, το είδος, την πλατφόρμα και το επίπεδο αξιολόγησης, να δημιουργούν και να διαχειρίζονται collections, να αφήνουν κριτικές, και να λαμβάνουν προτάσεις μέσω content-based και collaborative filtering.

Η αρχιτεκτονική της εφαρμογής στηρίζεται σε client–server μοντέλο, με ξεκάθαρο διαχωρισμό αρμοδιοτήτων:

- Frontend (Next.js): Υλοποιεί τη διεπαφή χρήστη, την πλοήγηση, τις σελίδες λιστών και προβολών, καθώς και την επικοινωνία με το backend μέσω REST API.
- Backend (Django + Django Rest Framework): Διαχειρίζεται την επιχειρησιακή λογική, την αυθεντικοποίηση χρηστών, τα μοντέλα δεδομένων και τα endpoints για CRUD λειτουργίες, αναζήτηση και συστήματα προτάσεων.
- Βάση Δεδομένων (PostgreSQL): Αποθηκεύει μόνιμα όλα τα δεδομένα, συμπεριλαμβανομένων παιχνιδιών, συλλογών, χρηστών και αξιολογήσεων, με σωστές σχέσεις Many-to-Many για genres, platforms και game modes.
- Docker & Docker Compose: Εξασφαλίζουν ομοιομορφία του περιβάλλοντος ανάπτυξης, εύκολη εγκατάσταση και τοπική εκτέλεση της εφαρμογής με μία εντολή.

Η επιλογή αυτών των τεχνολογιών επέτρεψε την ταυτόχρονη ανάπτυξη frontend και backend σε απομονωμένα, επεκτάσιμα περιβάλλοντα, ενώ η χρήση containerization εξασφαλίζει αναπαραγωγιμότητα και σταθερότητα.

5.2 Αξιολόγηση Επιτευγμάτων

Η υλοποίηση της πλατφόρμας πέτυχε τους βασικούς στόχους που είχαν τεθεί:

- Λειτουργικότητα: Οι χρήστες μπορούν να αναζητούν παιχνίδια με σύνθετα φίλτρα, να βλέπουν λεπτομέρειες παιχνιδιών, να δημιουργούν και να διαχειρίζονται collections, να αφήνουν κριτικές και να λαμβάνουν εξατομικευμένες προτάσεις.
- Εξατομίκευση: Η εφαρμογή υποστηρίζει content-based και collaborative filtering, προσφέροντας προτάσεις παιχνιδιών με βάση τις προτιμήσεις του χρήστη και τα κοινά χαρακτηριστικά άλλων χρηστών.

- Ασφάλεια και ευκολία πρόσβασης: Η ενσωμάτωση Google Authentication απλοποιεί τη διαδικασία σύνδεσης, διατηρώντας παράλληλα υψηλό επίπεδο ασφάλειας.
- Αναπαραγωγιμότητα και ανάπτυξη: Η χρήση Docker επέτρεψε τη σταθερή εκτέλεση τόσο σε περιβάλλον ανάπτυξης όσο και σε τοπικό περιβάλλον του χρήστη, χωρίς προβλήματα εξαρτήσεων ή διαμόρφωσης.

Παράλληλα, η πλατφόρμα υιοθέτησε βέλτιστες πρακτικές σε όλη τη διαχείριση δεδομένων, όπως pagination, prefetching και βελτιστοποιημένα queries, ώστε η εμπειρία του χρήστη να είναι ομαλή ακόμα και με μεγάλο όγκο δεδομένων.

5.3 Κριτική Αξιολόγηση

Η πλατφόρμα έδειξε ότι μπορεί να λειτουργεί αποδοτικά και να καλύπτει ένα μεγάλο μέρος των αναγκών των χρηστών. Συγκεκριμένα:

- Πλεονεκτήματα:
  - Η διαχείριση συλλογών και η σύνδεσή τους με συστήματα προτάσεων ενισχύει την ανακάλυψη νέων παιχνιδιών.
  - Η χρήση Django και DRF εξασφάλισε δομημένη ανάπτυξη και εύκολη συντήρηση του backend.
  - Το Next.js προσέφερε ταχύτητα, server-side rendering και responsive περιβάλλον για τον χρήστη.
- Προβλήματα και περιορισμοί:
  - Το cold-start problem για νέους χρήστες ή νέες συλλογές παραμένει περιορισμός για τις προτάσεις.
  - Η ακρίβεια του collaborative filtering περιορίζεται από τον αριθμό των διαθέσιμων χρηστών και collections.
  - Ορισμένα queries με πολλές σχέσεις Many-to-Many απαιτούν περαιτέρω optimization σε μεγάλα datasets.

Η παραπάνω αξιολόγηση βασίζεται σε παρατηρήσεις κατά τη διάρκεια των δοκιμών και στην εμπειρική χρήση της εφαρμογής, όπως επίσης και στη βιβλιογραφία για recommendation systems και web platforms.

5.4 Μαθήματα και Εμπειρίες

Κατά την ανάπτυξη της πλατφόρμας, αποκτήθηκαν σημαντικές γνώσεις και εμπειρίες:

- Η σημασία της σωστής οργάνωσης models, serializers και views στο Django για τη διατήρηση καθαρής αρχιτεκτονικής.
- Η ανάγκη για containerization και Docker για την ομοιομορφία περιβαλλόντων, που διευκολύνει την εγκατάσταση και εκτέλεση.
- Η πρακτική εφαρμογή recommendation systems και η αντιμετώπιση προβλημάτων όπως το filtering με πολλαπλά χαρακτηριστικά και η απόδοση queries.

- Η σημασία του user experience, όπου η σωστή πλοήγηση, responsive design και γρήγορη φόρτωση αποτελούν καθοριστικούς παράγοντες επιτυχίας μιας web εφαρμογής.

5.5 Περιορισμοί

Παρά τα σημαντικά επιτεύγματα, η πλατφόρμα έχει περιορισμούς που θα πρέπει να ληφθούν υπόψη:

- Ο περιορισμένος αριθμός χρηστών για δοκιμές μειώνει την αντιπροσωπευτικότητα των recommendations.
- Η εφαρμογή είναι αυτή τη στιγμή προσανατολισμένη σε τοπική εκτέλεση και δεν διαθέτει deployment σε live server.
- Περιορισμένες λειτουργίες ασφαλείας πέρα από Google Auth, όπως rate-limiting ή monitoring κακόβουλων αιτήσεων.
- Απουσία πλήρους υποστήριξης analytics και tracking για τη βελτιστοποίηση των προτάσεων με βάση τη συμπεριφορά χρήστη.

5.6 Προβλήματα κατά την ανάπτυξη και λύσεις τους

Κατά τη διάρκεια της ανάπτυξης της πλατφόρμας προέκυψαν ορισμένες τεχνικές προκλήσεις που απαιτούσαν προσεκτική προσέγγιση και βελτιστοποιήσεις για να διασφαλιστεί η ομαλή λειτουργία του συστήματος. Τα κυριότερα προβλήματα και οι τρόποι αντιμετώπισής τους περιγράφονται ακολούθως:

1. Σύνθετα queries με πολλά `ManyToMany` πεδία
   Η πλατφόρμα περιλαμβάνει οντότητες όπως παιχνίδια, genres, πλατφόρμες και game modes που σχετίζονται μεταξύ τους μέσω `ManyToMany` πεδίων. Αυτό είχε ως αποτέλεσμα η εκτέλεση ερωτημάτων για την ανάκτηση σύνθετων δεδομένων, π.χ. προτάσεων παιχνιδιών ή λιστών σε συλλογές, να είναι αργή όταν ο όγκος δεδομένων αυξανόταν.
   Η λύση βρέθηκε στη χρήση του Django `prefetch_related`, που επιτρέπει την ανάκτηση σχετικών αντικειμένων σε ένα μόνο query, μειώνοντας σημαντικά τον αριθμό των ερωτημάτων στη βάση δεδομένων και βελτιώνοντας την απόδοση. Επιπλέον, εφαρμόστηκαν τεχνικές φιλτραρίσματος και annotate για τον υπολογισμό similarity scores σε προτάσεις, ώστε να αποφευχθεί η εκτέλεση πολλών queries μέσα σε loops.
2. Συγχρονισμός frontend-backend σε σύνθετα φίλτρα αναζήτησης
   Η πλατφόρμα παρέχει λειτουργίες αναζήτησης με πολλαπλά κριτήρια, όπως είδος, πλατφόρμα, βαθμολογία και τίτλο παιχνιδιού. Ο συνδυασμός αυτών των φίλτρων απαιτούσε τη σωστή επικοινωνία μεταξύ frontend και backend, ώστε οι αλλαγές στα φίλτρα να αποτυπώνονται άμεσα στα αποτελέσματα.
   Η αντιμετώπιση έγινε με REST API endpoints που δέχονται παραμέτρους query string, σε συνδυασμό με διαχείριση κατάστασης στο frontend (state management) ώστε η αναζήτηση να ενημερώνεται δυναμικά χωρίς reload σελίδας. Η διαδικασία αυτή εξασφάλισε ομαλή και συνεπή εμπειρία χρήστη.

Συνολικά, η αντιμετώπιση των παραπάνω προβλημάτων συνέβαλε σημαντικά στη σταθερότητα, την απόδοση και την επεκτασιμότητα της πλατφόρμας, ενώ ταυτόχρονα απέδειξε την ανάγκη προσεκτικού σχεδιασμού των queries και της διαχείρισης του περιβάλλοντος ανάπτυξης.

5.7 Προτάσεις για Μελλοντική Επέκταση

Η πλατφόρμα διαθέτει ισχυρή βάση για μελλοντική ανάπτυξη, με δυνατότητες βελτίωσης και νέων λειτουργιών:

1. Προηγμένα recommendation systems: Εφαρμογή μηχανικής μάθησης για πιο ακριβείς και δυναμικές προτάσεις.
2. Mobile app ή PWA: Παροχή πρόσβασης μέσω κινητών συσκευών.
3. Analytics & personalization: Παρακολούθηση χρήσης και προσαρμογή της πλατφόρμας στις ανάγκες κάθε χρήστη.
4. Επέκταση μεταδεδομένων παιχνιδιών: Προσθήκη achievements, hours played, DLC metrics.
5. Live deployment και scaling: Φιλοξενία σε cloud, βελτιστοποίηση για πολλούς ταυτόχρονους χρήστες.
6. Βελτιώσεις ασφαλείας: Προστασία endpoints, rate limiting, και ολοκληρωμένο user management.

5.8 Συμπερασματική Αξιολόγηση

Η υλοποίηση της πλατφόρμας επιβεβαιώνει ότι μια καλά δομημένη web εφαρμογή με client-server αρχιτεκτονική, σύγχρονες τεχνολογίες και ενσωματωμένα recommendation systems μπορεί να προσφέρει πλήρη και εξατομικευμένη εμπειρία χρήστη. Η πλατφόρμα συνδυάζει τη λειτουργικότητα, την επεκτασιμότητα και την ασφάλεια, ενώ ταυτόχρονα αποτελεί ισχυρή βάση για περαιτέρω ανάπτυξη και επιστημονική μελέτη στον χώρο των συστημάτων προτάσεων για βιντεοπαιχνίδια.

Κεφάλαιο 6ο

Βιβλιογραφικές Πηγές

1. **Mozilla Developer Network.** *Introduction to the Fetch API.* https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch
2. **Next.js Documentation.** *Getting Started with Next.js.* https://nextjs.org/docs/getting-started
3. **React – A JavaScript library for building user interfaces.** *Main Concepts.* *https://legacy.reactjs.org/*
4. **TypeScript.** *TypeScript Documentation.* https://www.typescriptlang.org/docs/
5. **Tailwind CSS.** *Tailwind CSS Official Documentation.* https://tailwindcss.com/docs
6. **shadcn/ui.** *Component Library Documentation.* https://ui.shadcn.com/
7. **Django Project.** *Official Django Documentation.* https://docs.djangoproject.com/en/stable/
8. **Django REST framework.** *REST framework guide.* https://www.django-rest-framework.org/
9. **PostgreSQL Global Development Group.** *PostgreSQL Documentation.* https://www.postgresql.org/docs/
10. **IGDB (Internet Game Database).** *API Documentation.* https://api.igdb.com/
11. **Auth0 Documentation.** *OAuth 2.0 Introduction.* *https://auth0.com/docs*
12. **NextAuth.js.** *Next.js Authentication Library Documentation.* https://next-auth.js.org/getting-started/introduction
13. **Google Cloud.** *Using OAuth 2.0 to Access Google APIs.* https://developers.google.com/identity/protocols/oauth2
14. **Docker Documentation.** *Docker Overview.* https://docs.docker.com/get-started/overview/
15. **Docker Compose Documentation.** *Docker Compose CLI.* https://docs.docker.com/compose/reference/
16. **REST API Tutorial.** *RESTful API Principles.* https://restfulapi.net/
17. **JSON Web Tokens (JWT).** *Introduction to JWT.* https://jwt.io/introduction/
18. **OWASP.** *Authentication Cheat Sheet.* https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html

19. **Google Developers - Web Fundamentals.** *Responsive Design Basics.* https://developers.google.com/web/fundamentals/design-and-ux/responsive
20. **MDN Web Docs.** *Understanding HTTP Requests.* https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods
21. **Stanford CS 229: Machine Learning.** *Collaborative Filtering.* *https://cs229.stanford.edu/proj2013/Guthrie-YelpRecommendationSystem.pdf*
22. **Towards Data Science (Medium).** *Collaborative Filtering Explained.* *https://towardsdatascience.com/collaborative-filtering-simplified-the-basic-science-behind-recommendation-systems-1d7e7c58cd8/*
23. **Real Python.** *Build a Recommendation Engine with Collaborative Filtering.* *https://realpython.com/build-recommendation-engine-collaborative-filtering/*
24. **Medium.** *Detailed explanation about Collaborative filtering with Python examples.* *https://medium.com/@ichigo.v.gen12/detailed-explanation-about-collaborative-filtering-eab116e3b28b*
25. **Google Scholar - Recommendation Systems Survey.** *Recommender systems: an introduction.* https://scholar.google.com/scholar?q=recommender+systems+survey
26. **Stack Overflow Developer Survey.** *Developer Trends.* https://insights.stackoverflow.com/survey
27. **Web.Dev - Google.** *Progressive Web App (PWA) Fundamentals.* https://web.dev/progressive-web-apps/
28. **GitHub Docs.** *GitHub REST API.* https://docs.github.com/en/rest
29. **DigitalOcean.** *An Introduction to Proxies.* *https://www.digitalocean.com/community/conceptual-articles/introduction-to-proxies*
30. **Medium – CSS-Tricks.** *Responsive UI Patterns.* https://css-tricks.com/snippets/css/media-queries-for-standard-devices/
31. **Google Developers.** *Lighthouse - Performance Auditing.* https://developers.google.com/web/tools/lighthouse
32. **GitLab Docs.** *CI/CD Best Practices.* https://docs.gitlab.com/ee/ci/

Κεφάλαιο 7ο

Παραρτήματα

7.1 Backend

7.1.1 Βασικά Django Models

7.1.1.1 Genre Μοντέλο

Αντιπροσωπεύει τα είδη (genres) των παιχνιδιών, π.χ. "Action", "RPG", "Simulation".

Πεδίο	Τύπος	Περιγραφή
name	CharField(max_length=100, unique=True)	Το όνομα του είδους. Πρέπει να είναι μοναδικό.
description	TextField(blank=True, null=True)	Προαιρετική περιγραφή του είδους.

Παράδειγμα

```
{
  "id": 1,
  "name": "Action",
  "description": "Fast-paced games focused on reflexes and combat"
}
```

7.1.1.2 Platform Μοντέλο

Αντιπροσωπεύει τις πλατφόρμες (όπως PC, PlayStation, Xbox).

Πεδίο	Τύπος	Περιγραφή
name	CharField(max_length=100, unique=True)	Το όνομα της πλατφόρμας.
manufacturer	CharField(max_length=100, blank=True, null=True)	Προαιρετικό πεδίο για τον κατασκευαστή της πλατφόρμας.

Παράδειγμα

```
{
  "id": 1,
  "name": "PC",
  "manufacturer": "Various"
}
```


7.1.1.3 Developer Μοντέλο

Αντιπροσωπεύει τον δημιουργό (developer) ενός παιχνιδιού.

Πεδίο	Τύπος	Περιγραφή
name	CharField(max_length=255, unique=True)	Ονομασία του δημιουργού.
website	URLField(blank=True, null=True)	Επίσημη ιστοσελίδα του δημιουργού.

Παράδειγμα


```
{
  "id": 1,
  "name": "Rockstar Games",
  "website": "https://www.rockstargames.com/"
}
```


7.1.1.4 Publisher Μοντέλο

Αντιπροσωπεύει τον εκδότη (publisher) ενός παιχνιδιού.

Πεδίο	Τύπος	Περιγραφή
name	CharField(max_length=255, unique=True)	Ονομασία του εκδότη.
website	URLField(blank=True, null=True)	Ιστοσελίδα του εκδότη.

Παράδειγμα


```
{
```

```
  "id": 1,
  "name": "Rockstar Games",
  "website": "https://www.rockstargames.com/"
}
```

7.1.1.5 GameMode Μοντέλο

Κατηγοριοποιεί τα modes παιχνιδιού, όπως "Singleplayer", "Multiplayer", "Co-op".

Πεδίο	Τύπος	Περιγραφή
name	CharField(max_length=100, unique=True)	Ονομασία του mode.
description	TextField(blank=True, null=True)	Προαιρετική περιγραφή.

Παράδειγμα

```
{
  "id": 1,
  "name": "Single Player",
  "description": "Play alone against the game or AI"
}
```

7.1.1.6 Game Κύριο Μοντέλο

Αντιπροσωπεύει ένα παιχνίδι, περιέχοντας λεπτομέρειες όπως είδος, πλατφόρμες, βαθμολογίες και multimedia περιεχόμενο.

Κατηγορία	Πεδίο	Τύπος	Περιγραφή
Βασικά	title	CharField(max_length=255)	Τίτλος παιχνιδιού.
	description	TextField(blank=True, null=True)	Περιγραφή του παιχνιδιού.
	release_date	DateField(blank=True, null=True)	Ημερομηνία κυκλοφορίας.

	genre	ManyToManyField(Genre)	Συσχέτιση με πολλά είδη.
	platforms	ManyToManyField(Platform)	Διαθέσιμες πλατφόρμες.
	developer	ForeignKey(Developer)	Συσχέτιση με έναν δημιουργό.
	publisher	ForeignKey(Publisher)	Συσχέτιση με έναν εκδότη.
Στατιστικά	average_rating	DecimalField(max_digits=3, decimal_places=2)	Μέσος όρος αξιολογήσεων χρηστών.
	user_reviews_count	IntegerField	Σύνολο user reviews.
	critic_reviews_count	IntegerField	Σύνολο κριτικών από επαγγελματίες.
	popularity_score	DecimalField(max_digits=5, decimal_places=2)	Συνδυαστική βαθμολογία δημοφιλίας.
Multimedia	cover_image	URLField(blank=True, null=True)	Εξώφυλλο.
	trailer_url	URLField(blank=True, null=True)	Βίντεο τρέιλερ.
	official_website	URLField(blank=True, null=True)	Επίσημη ιστοσελίδα.
Λοιπά	created_at	DateTimeField(auto_now_add=True)	Ημερομηνία δημιουργίας εγγραφής.
	updated_at	DateTimeField(auto_now=True)	Τελευταία ενημέρωση.
	age_rating	CharField(max_length=10, blank=True, null=True)	PEGI/ESRB κωδικός ηλικίας.

	price	DecimalField(max_digits=6, decimal_places=2)	Τιμή παιχνιδιού.
	is_free_to_play	BooleanField(default=False)	Αν είναι free-to-play.
	dlc_count	IntegerField	Αριθμός διαθέσιμων DLCs.
	multiplayer	BooleanField(default=False)	Υποστήριξη multiplayer.
	game_modes	ManyToManyField(GameMode)	Υποστηριζόμενα game modes.

Παράδειγμα


```
{
  "id": 1,
  "title": "The Witcher 3: Wild Hunt",
  "description": "An action role-playing game set in an open world environment. You play as Geralt of Rivia, a monster hunter known as a Witcher, and search for your adopted daughter who is on the run from the Wild Hunt.",
  "release_date": "2015-05-19",
  "genre": [
    {
      "id": 2,
      "name": "Adventure",
      "description": "Story-driven games with exploration and puzzle-solving"
    },
    {
      "id": 3,
      "name": "RPG",
      "description": "Role-playing games with character development and storytelling"
    }
  ],
  "platforms": [
    {
      "id": 1,
      "name": "PC",
```

```
        "manufacturer": "Various"
    },
    {
        "id": 2,
        "name": "PlayStation 5",
        "manufacturer": "Sony"
    },
    {
        "id": 3,
        "name": "Xbox Series X",
        "manufacturer": "Microsoft"
    },
    {
        "id": 4,
        "name": "Nintendo Switch",
        "manufacturer": "Nintendo"
    },
    {
        "id": 5,
        "name": "PlayStation 4",
        "manufacturer": "Sony"
    },
    {
        "id": 6,
        "name": "Xbox One",
        "manufacturer": "Microsoft"
    }
],
"developer": {
    "id": 2,
    "name": "CD Projekt Red",
    "website": "https://www.cdprojektred.com/"
},
"publisher": {
    "id": 2,
    "name": "CD Projekt",
    "website": "https://www.cdprojekt.com/"
},
"average_rating": "4.80",
```

```
  "user_reviews_count": 1250,
  "critic_reviews_count": 87,
  "popularity_score": "9.50",
  "cover_image":
"https://upload.wikimedia.org/wikipedia/en/0/0c/Witcher_3_cover_art.jpg",
  "trailer_url": "https://www.youtube.com/watch?v=c0i88t0Kacs",
  "official_website": "https://thewitcher.com/en/witcher3",
  "created_at": "2023-01-01T18:00:00-06:00",
  "updated_at": "2023-01-01T18:00:00-06:00",
  "age_rating": "M",
  "price": "39.99",
  "is_free_to_play": false,
  "dlc_count": 2,
  "multiplayer": false,
  "game_modes": [
    {
      "id": 1,
      "name": "Single Player",
      "description": "Play alone against the game or AI"
    },
    {
      "id": 7,
      "name": "Story Mode",
      "description": "Follow a narrative-driven campaign"
    }
  ]
}
```


7.1.1.7 Collection **Μοντέλο**

Επιτρέπει στους χρήστες να δημιουργούν συλλογές από παιχνίδια.

Πεδίο	Τύπος	Περιγραφή
title	CharField(max_length=255)	Τίτλος συλλογής.
description	TextField(blank=True, null=True)	Περιγραφή.
games	ManyToManyField(Game)	Παιχνίδια της συλλογής.

created_at	DateTimeField(auto_now_add=True)	Ημερομηνία δημιουργίας.
updated_at	DateTimeField(auto_now=True)	Τελευταία ενημέρωση.

Παράδειγμα

```
{
  "id": 1,
  "title": "Best RPGs of All Time",
  "description": "A collection of the most influential RPGs that defined the genre.",
  "games": [...]
  "created_at": "2023-01-31T18:00:00-06:00",
  "updated_at": "2023-01-31T18:00:00-06:00"
}
```

7.1.1.8 Review Μοντέλο

Αντιπροσωπεύει αξιολογήσεις από χρήστες για κάθε παιχνίδι. Κάθε χρήστης μπορεί να βαθμολογήσει ένα παιχνίδι μόνο μία φορά.

Πεδίο	Τύπος	Περιγραφή
game	ForeignKey(Game)	Το παιχνίδι που αξιολογείται.
user	ForeignKey(User)	Ο χρήστης που κάνει την αξιολόγηση.
rating	PositiveSmallIntegerField	Βαθμολογία (συνήθως 1-5).
comment	TextField(blank=True)	Σχόλιο χρήστη.
created_at	DateTimeField(auto_now_add=True)	Ημερομηνία υποβολής.

Meta:

- unique_together = ("game", "user") → Κάθε χρήστης μπορεί να κάνει μία κριτική ανά παιχνίδι.
- ordering = ["-created_at"] → Οι πιο πρόσφατες κριτικές πρώτες.

Παράδειγμα


```
{
  "id": 1,
  "game": 1,
  "user": 1,
  "user_username": "admin",
  "rating": 5,
  "comment": "The best game I've ever played!",
  "created_at": "2025-07-20T10:53:42.829071-05:00"
}
```


7.1.1.9 Screenshot Μοντέλο

Αντιπροσωπεύει screenshots (στιγμιότυπα οθόνης) για κάθε παιχνίδι.

Πεδίο	Τύπος	Περιγραφή
game	ForeignKey(Game)	Συσχετισμένο παιχνίδι.
image_url	URLField(blank=True, null=True)	URL της εικόνας.

7.1.2 Τεκμηρίωση REST API

7.1.2.1 GenreViewSet

Διαχειρίζεται τα είδη (genres) των παιχνιδιών. Παρέχει πλήρες CRUD API και εμφανίζεται στο frontend για φιλτράρισμα και οργάνωση παιχνιδιών.

Μέθοδος	URL	Περιγραφη
GET	/api/genres/	Λίστα όλων των genres
POST	/api/genres/	Δημιουργία νέου genre
PUT	/api/genres/{id}/	Ενημέρωση genre
DELETE	/api/genres/{id}/	Διαγραφή genre

Απόσπασμα κώδικα:

```
class GenreViewSet(viewsets.ModelViewSet):
  queryset = Genre.objects.all()
  serializer_class = GenreSerializer
  pagination_class = NoPagination
```

7.1.2.2 PlatformViewSet

Διαχειρίζεται τις πλατφόρμες (PC, PlayStation, Xbox) και παρέχει CRUD λειτουργίες για τις πλατφόρμες του συστήματος.

Μέθοδος	**URL**	**Περιγραφη**
GET	/api/platforms/	Λίστα πλατφορμών
POST	/api/platforms/	Δημιουργία νέας πλατφόρμας
PUT	/api/platforms/{id}/	Ενημέρωση πλατφόρμας
DELETE	/api/platforms/{id}/	Διαγραφή πλατφόρμας

Απόσπασμα κώδικα:

```
class PlatformViewSet(viewsets.ModelViewSet):
  queryset = Platform.objects.all()
  serializer_class = PlatformSerializer
  pagination_class = NoPagination
```

7.1.2.3 DeveloperViewSet

Διαχειρίζεται τους developers των παιχνιδιών και παρέχει CRUD λειτουργίες για την οντότητα Developer.

Μέθοδος	**URL**	**Περιγραφη**
GET	/api/developers/	Λίστα όλων των developers
POST	/api/developers/	Δημιουργία νέου developer
PUT	/api/developers/{id}/	Ενημέρωση developer

DELETE	/api/developers/{id}/	Διαγραφή developer

Απόσπασμα κώδικα:

```
class DeveloperViewSet(viewsets.ModelViewSet):
  queryset = Developer.objects.all()
  serializer_class = DeveloperSerializer

  def create(self, request, *args, **kwargs):
     serializer = self.get_serializer(data=request.data)
     serializer.is_valid(raise_exception=True)
     self.perform_create(serializer)
     return Response(serializer.data, status=status.HTTP_201_CREATED)
```

7.1.2.4 PublisherViewSet

Διαχειρίζεται τους publishers των παιχνιδιών και παρέχει CRUD λειτουργίες για την οντότητα Publisher.

Μέθοδος	**URL**	**Περιγραφη**
GET	/api/publishers/	Λίστα όλων των publishers
POST	/api/publishers/	Δημιουργία νέου publisher
PUT	/api/publishers/{id}/	Ενημέρωση publisher
DELETE	/api/publishers/{id}/	Διαγραφή publisher

Απόσπασμα κώδικα:

```
class PublisherViewSet(viewsets.ModelViewSet):
  queryset = Publisher.objects.all()
  serializer_class = PublisherSerializer

  def create(self, request, *args, **kwargs):
     serializer = self.get_serializer(data=request.data)
```

```
    serializer.is_valid(raise_exception=True)
    self.perform_create(serializer)
    return Response(serializer.data, status=status.HTTP_201_CREATED)
```

7.1.2.5 GameModeViewSet

Διαχειρίζεται τα game modes των παιχνιδιών και παρέχει CRUD λειτουργίες για την οντότητα GameMode.

Μέθοδος	**URL**	**Περιγραφη**
GET	/api/game-modes/	Λίστα όλων των game modes
POST	/api/game-modes/	Δημιουργία νέου game mode
PUT	/api/game-modes/{id}/	Ενημέρωση game mode
DELETE	/api/game-modes/{id}/	Διαγραφή game mode

Απόσπασμα κώδικα:

```
class GameModeViewSet(viewsets.ModelViewSet):
  queryset = GameMode.objects.all()
  serializer_class = GameModeSerializer

  def create(self, request, *args, **kwargs):
    serializer = self.get_serializer(data=request.data)
    serializer.is_valid(raise_exception=True)
    self.perform_create(serializer)
    return Response(serializer.data, status=status.HTTP_201_CREATED)
```

7.1.2.6 GameViewSet

Το GameViewSet παρέχει πλήρη διαχείριση των παιχνιδιών, καθώς και **custom endpoints** για:

- Featured, Popular, Recent games
- Αναζήτηση με φίλτρα
- Προβολή screenshots και reviews
- Similar games based on genre

Endpoint	**Περίγραφη**
/games/featured	Παιχνίδια με υψηλή βαθμολογία (average_rating > 4.0)
/games/popular	Δημοφιλέστερα παιχνίδια (popularity_score)
/games/recent	Πρόσφατες κυκλοφορίες
/games/{id}/similar	Παρόμοια παιχνίδια με κοινά genres
/games/{id}/reviews	Κριτικές για το παιχνίδι
/games/{id}/screenshots	Screenshots του παιχνιδιού
/games/by-genre?genre_id=X	Φιλτράρισμα παιχνιδιών ανά genre

Σημαντικό απόσπασμα custom action:

```
@action(detail=False, methods=["get"], url_path="popular")
def popular_games(self, request):
  games = self.get_queryset().order_by("-popularity_score")[:10]
  return Response(self.get_serializer(games, many=True).data)
```

7.1.2.7 CollectionViewSet

Διαχειρίζεται τις συλλογές (collections) των χρηστών, επιτρέποντας CRUD λειτουργίες και προσθήκη/αφαίρεση παιχνιδιών. Υποστηρίζει επίσης recommendation endpoints.

Endpoint	**Περιγραφή**
/collections/by_user/{user_id}	Λίστα collections ενός χρήστη
/collections/{id}/games	Προσθήκη παιχνιδιού σε collection
/collections/{id}/games/{game_id}	Αφαίρεση παιχνιδιού από collection
/collections/{id}/recommendations	Content-based recommendations

/collections/{id}/recommendations-collab	Collaborative filtering recommendations

Σημαντικό απόσπασμα:

```
@action(detail=True, methods=["post"], url_path="games")
def add_game(self, request, pk=None):
  collection = self.get_object()
  game_id = request.data.get("game_id")
  collection.games.add(game_id)
  return Response({"status": "game added"}, status=status.HTTP_200_OK)
```

7.1.2.8 ReviewViewSet

Διαχειρίζεται τις κριτικές παιχνιδιών και εξασφαλίζει ότι κάθε χρήστης μπορεί να αφήσει μία μόνο κριτική ανά παιχνίδι.

Σημαντικό απόσπασμα:

```
def perform_create(self, serializer):
  game = serializer.validated_data.get("game")
  user = self.request.user
  if Review.objects.filter(game=game, user=user).exists():
    return Response({"detail": "Already reviewed"}, status=400)
  serializer.save(user=user)
```

7.1.3 Συστήματα Προτάσεων

7.1.3.1 Content-Based Filtering

- Λαμβάνονται χαρακτηριστικά των παιχνιδιών σε μια συλλογή: genres, platforms, game modes, developer, publisher
- Υπολογίζεται **similarity score** για κάθε παιχνίδι που δεν υπάρχει ήδη στη συλλογή
- Ταξινόμηση κατά similarity score, δημοφιλία και μέση βαθμολογία

Απόσπασμα κώδικα:

```
similarity_score = count_common_genres + count_common_platforms +
count_common_modes
similar_games =
Game.objects.exclude(id__in=collection_games).order_by("-similarity_score")[:limit]
```

7.1.3.2 Collaborative Filtering

- Βρίσκει άλλες συλλογές χρηστών που έχουν κοινά παιχνίδια με τη συλλογή στόχο
- Υπολογίζει overlap και προτείνει νέα παιχνίδια που υπάρχουν σε αυτές τις συλλογές
- Ταξινόμηση των παιχνιδιών με βάση το overlap

Απόσπασμα κώδικα:

```
for coll in other_collections:
  overlap = len(target_games ∩ coll.games)
  if overlap > 0:
    for game in coll.games - target_games:
      similarity_scores[game] += overlap

recommended_games = top(similarity_scores, limit=10)
```

7.2 Frontend

7.2.1 Αυθεντικοποίηση Χρηστών με NextAuth

Το παρόν παράρτημα παρουσιάζει την υλοποίηση της αυθεντικοποίησης χρηστών στο frontend της εφαρμογής, η οποία βασίζεται στη βιβλιοθήκη NextAuth και στο πρωτόκολλο OAuth 2.0 μέσω του παρόχου Google.

Η διαδικασία σύνδεσης πραγματοποιείται μέσω εξωτερικού παρόχου ταυτότητας, εξασφαλίζοντας ασφαλή αυθεντικοποίηση χωρίς την αποθήκευση ευαίσθητων διαπιστευτηρίων στην εφαρμογή. Η διαχείριση της συνεδρίας υλοποιείται με στρατηγική JWT, επιτρέποντας την αποθήκευση του access token του παρόχου εντός του JSON Web Token.

Το access token ενσωματώνεται στη συνεδρία του χρήστη και καθίσταται διαθέσιμο στο frontend, ώστε να μπορεί να χρησιμοποιηθεί για την εξουσιοδοτημένη επικοινωνία με το backend σύστημα. Με τον τρόπο αυτό επιτυγχάνεται ενιαία και ασφαλής ταυτοποίηση του χρήστη σε όλα τα επίπεδα της εφαρμογής.

Η συγκεκριμένη προσέγγιση επιτρέπει τον διαχωρισμό της ευθύνης αυθεντικοποίησης από τη λογική του backend, ενώ παράλληλα διευκολύνει την επεκτασιμότητα της εφαρμογής με επιπλέον παρόχους ταυτότητας στο μέλλον.

Απόσπασμα κώδικα:

```
import NextAuth, { NextAuthOptions } from "next-auth";
import GoogleProvider from "next-auth/providers/google";

export const authOptions: NextAuthOptions = {
 providers: [
  GoogleProvider({
   clientId: process.env.GOOGLE_CLIENT_ID!,
   clientSecret: process.env.GOOGLE_CLIENT_SECRET!,
  }),
 ],

 pages: {
  signIn: "/auth",
 },

 secret: process.env.NEXTAUTH_SECRET,

 session: {
  strategy: "jwt",
 },

 callbacks: {
  async jwt({ token, account }) {
   if (account?.access_token) {
    token.accessToken = account.access_token;
   }
```

```
    return token;
   },

   async session({ session, token }) {
    session.accessToken = token.accessToken as string;
    return session;
   },
  },
};

export default NextAuth(authOptions);
```

7.2.2 Επικοινωνία με Backend μέσω Axios

Το παρόν παράρτημα περιγράφει τον μηχανισμό επικοινωνίας του frontend με το backend της εφαρμογής μέσω της βιβλιοθήκης Axios, η οποία χρησιμοποιείται για την αποστολή HTTP αιτημάτων προς το REST API.

Δημιουργείται κεντρικό instance του Axios με κοινή βασική διεύθυνση (base URL), ώστε να διασφαλίζεται ομοιομορφία και επαναχρησιμοποίηση σε όλα τα αιτήματα της εφαρμογής. Μέσω request interceptors, κάθε αίτημα εμπλουτίζεται αυτόματα με το access token του χρήστη, το οποίο προστίθεται στο HTTP header Authorization με τη μορφή Bearer token.

Η προσέγγιση αυτή επιτρέπει την υλοποίηση εξουσιοδοτημένων αιτημάτων προς το backend χωρίς την ανάγκη επανάληψης λογικής σε κάθε μεμονωμένο request. Παράλληλα, μέσω response interceptors, διαχειρίζονται κεντρικά σφάλματα εξουσιοδότησης και μη διαθέσιμων πόρων, εξασφαλίζοντας συνεπή συμπεριφορά της εφαρμογής σε περιπτώσεις σφαλμάτων.

Η συγκεκριμένη αρχιτεκτονική βελτιώνει τη συντηρησιμότητα του κώδικα, ενισχύει την ασφάλεια της επικοινωνίας και διαχωρίζει καθαρά τη λογική αυθεντικοποίησης από την επιχειρησιακή λογική της εφαρμογής.

Απόσπασμα κώδικα:

```
import axios, { AxiosError, AxiosResponse } from "axios";
import { getSession } from "next-auth/react";

const backendUrl = process.env.NEXT_PUBLIC_BACKEND_URL;
```

```
export const axiosInstance = axios.create({
  baseURL: `${backendUrl}/api`,
  headers: {
    "Content-Type": "application/json",
  },
});

axiosInstance.interceptors.request.use(
  async (config) => {
    if (typeof window !== "undefined") {
      const session = await getSession();

      if (session?.accessToken) {
        config.headers.Authorization = `Bearer ${session.accessToken}`;
      }
    }

    return config;
  },
  (error) => Promise.reject(error)
);

axiosInstance.interceptors.response.use(
  (response: AxiosResponse) => response,
  async (error: AxiosError) => {
    if (error.response?.status === 403) {
      window.location.href = "/auth";
    }

    if (error.response?.status === 404) {
      return false;
    }

    return Promise.reject(error);
```

```
  }
);
```

7.2.3 Σελίδα Προβολής Πληροφοριών Βιντεοπαιχνιδιού

Το παρόν παράρτημα παρουσιάζει ενδεικτικά τη σελίδα προβολής λεπτομερειών ενός παιχνιδιού, η οποία αποτελεί βασικό σημείο αλληλεπίδρασης του χρήστη με την εφαρμογή. Η σελίδα υλοποιείται ως δυναμική διαδρομή και φορτώνει δεδομένα με βάση το αναγνωριστικό του παιχνιδιού που παρέχεται μέσω του URL.

Η ανάκτηση των δεδομένων πραγματοποιείται μέσω εξειδικευμένων custom hooks, τα οποία επικοινωνούν με το backend REST API για τη φόρτωση των βασικών πληροφοριών του παιχνιδιού, των αξιολογήσεων χρηστών, του σχετικού πολυμεσικού περιεχομένου και των προτεινόμενων παρόμοιων παιχνιδιών. Με τον τρόπο αυτό επιτυγχάνεται διαχωρισμός της λογικής ανάκτησης δεδομένων από τη λογική παρουσίασης, βελτιώνοντας τη συντηρησιμότητα του κώδικα.

Η κατάσταση της εφαρμογής (state) διαχειρίζεται τόσο μέσω των μηχανισμών των hooks όσο και μέσω τοπικού state, το οποίο χρησιμοποιείται για τη διαχείριση αλληλεπιδράσεων του χρήστη, όπως η προσθήκη ενός παιχνιδιού σε συλλογή. Η αλληλεπίδραση αυτή ενεργοποιεί αντίστοιχες κλήσεις προς το backend, διασφαλίζοντας τη συγχρονισμένη ενημέρωση των δεδομένων.

Η σελίδα συνδυάζει δεδομένα προερχόμενα από διαφορετικές πηγές του backend και τα παρουσιάζει με συνεκτικό τρόπο, υποστηρίζοντας βασικές λειτουργίες της πλατφόρμας όπως η αξιολόγηση περιεχομένου, η δημιουργία συλλογών και η παροχή εξατομικευμένων προτάσεων. Η συγκεκριμένη υλοποίηση αποτυπώνει τη συνολική αρχιτεκτονική του frontend και τον τρόπο με τον οποίο αυτό αλληλεπιδρά με το backend σύστημα.

Απόσπασμα κώδικα:

```
"use client";
import { useRouter } from "next/router";
import Image from "next/image";
import Link from "next/link";
import Layout from "@/components/layout/layout";
import GameDetailsTabs from "@/components/games/game-details-tabs";
import GameInfoSidebar from "@/components/games/game-info-sidebar";
import { Button } from "@/components/ui/button";
import { Badge } from "@/components/ui/badge";
import { Heart } from "lucide-react";
```

```
import RatingStars from "@/components/ui/rating-stars";
import { useGame } from "@/lib/hooks/useGames";
import { useSimilarGames } from "@/lib/hooks/useSimilarGames";
import { useGameReviews } from "@/lib/hooks/useGamesReviews";
import { useGameScreenshots } from "@/lib/hooks/useGamesScreenshots";
import { useState } from "react";
import AddToCollectionPopup from "@/components/collections/add-to-collection";
import { useAuth } from "@/lib/hooks/useAuth";

export default function GameDetailPage() {
  const router = useRouter();
  const { id } = router.query;
  const gameId = typeof id === "string" ? Number(id) : undefined;

  const [showPopup, setShowPopup] = useState(false)
  const { user } = useAuth()

  const { data: game, isLoading: loadingGame } = useGame(gameId);
  const { data: similarGames } = useSimilarGames(gameId);
  const { data: reviews } = useGameReviews(gameId);
  const { data: screenshots } = useGameScreenshots(gameId);

  if (loadingGame) return <Layout title="Loading...">Loading…</Layout>;
  if (!game) return <Layout title="Game Not Found">Not found</Layout>;

  if (!game && typeof id !== "undefined") {
    return (
      <Layout title="Game Not Found">
        <div className="container mx-auto px-4 py-16 text-center">
          <h1 className="text-3xl font-bold mb-4">Game Not Found</h1>
          <p className="mb-8">
            The game you are looking for doesnt exist or has been
            removed.
          </p>
```

```
            <Link href="/games">
              <Button>Browse All Games</Button>
            </Link>
          </div>
        </Layout>
    );
}

if (!game) {
    return (
        <Layout title="Loading...">
          <div className="container mx-auto px-4 py-16 text-center">
            <h1 className="text-3xl font-bold mb-4">Loading...</h1>
          </div>
        </Layout>
    );
}

return (
    <Layout
        title={game.title}
        description={
          game.description?.substring(0, 160) ||
          `Details about ${game.title}`
        }
    >
        <div className="relative h-[400px] bg-gradient-to-r from-gray-900 to-gray-800">
          {game.cover_image && (
            <div className="absolute inset-0 opacity-30">
              <Image
                src={game.cover_image || "/placeholder.svg"}
                alt={game.title}
                fill
                className="object-cover object-center"
```

```
                priority
              />
            </div>
          )}

          <div className="container mx-auto px-4 h-full relative z-10">
            <div className="flex flex-col md:flex-row items-end h-full pb-8 gap-8">
              <div className="w-40 h-56 md:w-48 md:h-64 flex-shrink-0 rounded-lg
overflow-hidden shadow-lg mt-auto bg-gray-800">
                <Image
                  src={
                    game.cover_image ||
                    "/placeholder.svg?height=300&width=200"
                  }
                  alt={game.title}
                  width={200}
                  height={300}
                  className="w-full h-full object-cover"
                />
              </div>

              <div className="flex-grow">
                <div className="flex flex-wrap gap-2 mb-2">
                  {game.genre.map((g) => (
                    <Badge
                      key={g.id}
                      variant="secondary"
                      className="bg-emerald-500/20 text-emerald-300
hover:bg-emerald-500/30"
                    >
                      {g.name}
                    </Badge>
                  ))}
                </div>
```

```
<h1 className="text-4xl md:text-5xl font-bold text-white mb-2">
  {game.title}
</h1>

<div className="flex items-center gap-4 mb-4">
  <div className="flex items-center">
    <RatingStars rating={game.average_rating} />
    <span className="ml-2 text-white">
      {game.average_rating}
    </span>
  </div>
  <span className="text-gray-300">|</span>
  <span className="text-gray-300">
    {game.release_date}
  </span>
  {game.age_rating && (
    <>
      <span className="text-gray-300">|</span>
      <span className="px-2 py-1 bg-gray-700 text-white text-xs rounded">
        {game.age_rating}
      </span>
    </>
  )}
</div>

<div className="flex flex-wrap gap-3">
  {user && (
    <>
    <Button
    className="bg-emerald-500 hover:bg-emerald-600"
    onClick={() => setShowPopup(true)}
    >
    <Heart className="mr-2 h-4 w-4" /> Add to Collection
```

```
                </Button>
                {showPopup && (
                <AddToCollectionPopup
                    gameId={game.id}
                    onClose={() => setShowPopup(false)}
                />
                )}
            </>
            )}
            {game.trailer_url && (
                <Link
                    href={game.trailer_url}
                    target="_blank"
                    rel="noopener noreferrer"
                >
                    <Button
                        variant="outline"
                        className="border-white text-white hover:bg-white/10"
                    >
                        Watch Trailer
                    </Button>
                </Link>
            )}
            {game.official_website && (
                <Link
                    href={game.official_website}
                    target="_blank"
                    rel="noopener noreferrer"
                >
                    <Button
                        variant="outline"
                        className="border-white text-white hover:bg-white/10"
                    >
                        Official Website
```

```
                    </Button>
                  </Link>
                )}
              </div>
            </div>
          </div>
        </div>

        <div className="container mx-auto px-4 py-8">
          <div className="grid grid-cols-1 lg:grid-cols-3 gap-8">
            {/* Main Content */}
            <div className="lg:col-span-2">
              <GameDetailsTabs
                game={game}
                similarGames={similarGames || []}
                reviews={reviews}
                screenshots={screenshots}
              />
            </div>

            <div>
              <GameInfoSidebar game={game} />
            </div>
          </div>
        </div>
      </Layout>
    );
}
```

7.2.4 React Query / Λογική Ανάκτησης Δεδομένων

Το παρόν παράρτημα παρουσιάζει ενδεικτικά τον τρόπο με τον οποίο η εφαρμογή υλοποιεί τη λογική ανάκτησης δεδομένων χρησιμοποιώντας τη βιβλιοθήκη React Query. Η

χρήση της επιτρέπει την αποτελεσματική διαχείριση αιτήσεων προς το backend, τη διατήρηση cache, καθώς και την παρακολούθηση των καταστάσεων φόρτωσης και σφαλμάτων.

Στην υλοποίηση χρησιμοποιούνται custom hooks, όπως το useFeaturedGames, το οποίο είναι υπεύθυνο για την ανάκτηση της λίστας των επιλεγμένων παιχνιδιών. Το hook βασίζεται στο useQuery της React Query και επικοινωνεί με το REST API μέσω ενός προκαθορισμένου axios instance.

Απόσπασμα κώδικα:

```
import { useQuery } from '@tanstack/react-query'
import { axiosInstance as axios } from "@/lib/axios-config";

export const useFeaturedGames = () => {
 return useQuery({
  queryKey: ['featured-games'],
  queryFn: async () => {
   const res = await axios.get('/games/featured')
   return res.data
  },
 })
}
```

7.3 Docker Compose

Το παρόν παράρτημα παρουσιάζει τη χρήση του Docker Compose για την ανάπτυξη και διαχείριση των υπηρεσιών της εφαρμογής, καλύπτοντας frontend, backend και βάση δεδομένων. Το Compose επιτρέπει την εκτέλεση πολλών containers με ένα ενιαίο configuration, διασφαλίζοντας ότι όλα τα στοιχεία της εφαρμογής μπορούν να τρέχουν απομονωμένα αλλά και συνδεδεμένα μεταξύ τους.

Στην υλοποίηση αυτή:

Η υπηρεσία web_app_frontend αντιπροσωπεύει το frontend της εφαρμογής (Next.js/React). Τρέχει σε container και εκθέτει την θύρα 3000 για πρόσβαση από τον browser. Τα volumes επιτρέπουν την ανάγνωση και επεξεργασία του κώδικα σε πραγματικό χρόνο. Η εξάρτησή της από το backend καθορίζεται μέσω depends_on.

Η υπηρεσία webappbackend αντιπροσωπεύει το backend (Django), εκθέτει τις απαραίτητες θύρες για την επικοινωνία με το frontend και άλλα εργαλεία, ενώ τα volumes επιτρέπουν live development. Οι μεταβλητές περιβάλλοντος (env_file) χρησιμοποιούνται για την ασφαλή ρύθμιση των allowed hosts και άλλων παραμέτρων.

Η υπηρεσία web_app_db χρησιμοποιεί PostgreSQL για τη βάση δεδομένων. Τα δεδομένα αποθηκεύονται σε named volumes για επιμονή μεταξύ επανεκκινήσεων.

Το δίκτυο games-app-net επιτρέπει την επικοινωνία των containers μεταξύ τους χωρίς να εκτίθενται άσκοπα στο εξωτερικό περιβάλλον.

Η εντολή depends_on καθορίζει τη σειρά εκκίνησης των υπηρεσιών, εξασφαλίζοντας ότι η βάση δεδομένων είναι διαθέσιμη πριν ξεκινήσει το backend και ότι το backend είναι διαθέσιμο πριν το frontend.

Η χρήση του Docker Compose παρέχει μια αναπαραγώγιμη και καθαρή ανάπτυξη για όλα τα μέλη της ομάδας και τους εξεταστές, επιτρέποντας την εύκολη εκκίνηση της εφαρμογής με μία εντολή, χωρίς σύνθετες τοπικές ρυθμίσεις.

Απόσπασμα κώδικα:

```yaml
version: "3"

networks:
  games-app-net:

services:

  web_app_frontend:
    build:
      context: ./frontend
      dockerfile: ./compose/local/Dockerfile
    volumes:
      - ./frontend:/app
      - /app/node_modules
      - /app/.next
    ports:
      - "3000:3000"
    networks:
      - games-app-net
```

```
  env_file:
    - ./frontend/.env.local
  depends_on:
    - webappbackend

webappbackend:
  build:
    context: ./backend
    dockerfile: ./compose/local/Dockerfile
  command: /start.sh
  volumes:
    - ./backend:/usr/src/games_app_backend/
  ports:
    - "8001:8000"
  networks:
    - games-app-net
  env_file:
    - ./backend/.env/local/.env_app
  environment:
    - DJANGO_ALLOWED_HOSTS=web_app_backend,web_app_frontend,localhost,127.0.0.1
  depends_on:
    - web_app_db

web_app_db:
  image: postgres:12
  volumes:
    - games_app_backend_dev_data:/var/lib/postgresql/data/
  ports:
    - "6001:5432"
  networks:
    - games-app-net
  env_file:
    - ./backend/.env/local/.env_db
```

```
volumes:
  games_app_backend_dev_data: {}
```