

[Web Game Showcase] [Web αλληλεπιδραστική εφαρμογή παρουσίασης και επιλογής ψηφιακών παιχνιδιών]

[Σπύρος Καφίρης Π15055]

Επιβλέπουσα: κ.Κωνσταντίνα Χρυσafiάδη

Τριμελής επιτροπή:

Χρυσafiάδη Κωνσταντίνα, Ε.Δι.Π.,

Μαρία Βίρβου, Καθηγήτρια,

Ευθύμιος Αλέπης, Αναπληρωτής Καθηγητής

[Περίληψη]

Στις μέρες μας, όλοι χρειαζόμαστε ελεύθερο χρόνο για να ανταπεξέλθουμε στον γρήγορο ρυθμό της καθημερινότητας. Για να είναι παραγωγικοί, πολλοί άνθρωποι βρίσκουν κάποια ενασχόληση άσχετη με τον επαγγελματικό τους προσανατολισμό και το περιβάλλον τους. Ένα χόμπι με πάνω από 3 δισεκατομμύριους ενασχολούμενους είναι το Gaming. Άνθρωποι ψυχαγωγούνται ή και κάνουν επάγγελμα την αγαπημένη τους ασχολία είτε αυτή αφορά υπολογιστή, κάποια κονσόλα ή το κινητό τους τηλέφωνο. Παιχνίδια πλέον υπάρχουν μέχρι και σε ένα ρολόι χειρός.

Ανάλογα τον τύπο ανθρώπου, κάποιος μπορεί να αφιερώσει πολλές ώρες βλέποντας κριτικές παιχνιδιών μέχρι να αποφασίσει που θα αφιερώσει τον περιορισμένο ελεύθερο χρόνο του. Προσθέτοντας παιχνίδια που του τράβηξαν το ενδιαφέρον σε μία λίστα, αποθηκεύοντας αγαπημένους τίτλους στους σελιδοδείκτες του, ή γεμίζοντας τις σημειώσεις του κινητού του η επιλογή των κατάλληλων παιχνιδιών μπορεί να πολύωρη ασχολία για κάποιον πριν καν αρχίσει να απολαμβάνει τον ελεύθερο χρόνο του.

Σε αυτό ακριβώς το κομμάτι έρχεται να βοηθήσει αυτή η εφαρμογή. Ένα μέρος όπου συγκεντρώνονται πολλοί τίτλοι παιχνιδιών, μαζί με τις πληροφορίες που χρειάζεται κανείς για το κάθε παιχνίδι σε ένα ατμοσφαιρικό περιβάλλον. Με εύχρηστες και απλές επιλογές για να μπορεί να βρει ο χρήστης ακριβώς αυτό που ψάχνει και επιπλέον να μπορεί να αποθηκεύσει το κάθε παιχνίδι που τον ενδιαφέρει σε μια μοναδική σελίδα αγαπημένων που αποθηκεύει όλες τις χρήσιμες πληροφορίες για τα παιχνίδια που έχει επιλέξει.

[Abstract]

Nowadays, we all need some free time to cope with the fast pace of everyday life. To be productive, many people find some occupation unrelated to their professional orientation and environment. A hobby with over 3 billion people involved is Gaming. People entertain themselves or make a profession of their favorite occupation, whether it concerns a computer, a console or their mobile phone. Games now even exist on a wristwatch.

Depending on the type of person, someone can spend hours looking at game reviews before deciding where to spend their limited free time. Adding games that caught his interest to a list, saving favorite titles in his bookmarks, or filling up his mobile notes, choosing the right games can be a long-term occupation for someone before they even start to enjoy their free time.

This is exactly where this app comes in to help. A place where many game titles are gathered together with the information one needs about each game in an atmospheric environment. With easy-to-use and simple options so that the user can find exactly what he is looking for and in addition to be able to save every game he is interested in on a unique favorites page that stores all the useful information about the games he has chosen.

[Περίληψη]..... 1

[Abstract]	1
[Στόχοι]	3
[Παρόμοιες Εφαρμογές]	3
[Περιγραφή εφαρμογής]	5
[Ανάλυση απαιτήσεων]	5
[Οδηγός Εγκατάστασης]	5
[Σχεδιασμός]	12
[Τεχνολογίες που χρησιμοποιήθηκαν]	12
{Back-end}.....	13
{Βάση Δεδομένων- TablePlus}.....	19
{Front-end}.....	20
[Web Game Showcase τελική εφαρμογή]	28
[Μελλοντικές λειτουργίες που θα προστεθούν]	30
[Συμπεράσματα]	31

[Στόχοι]

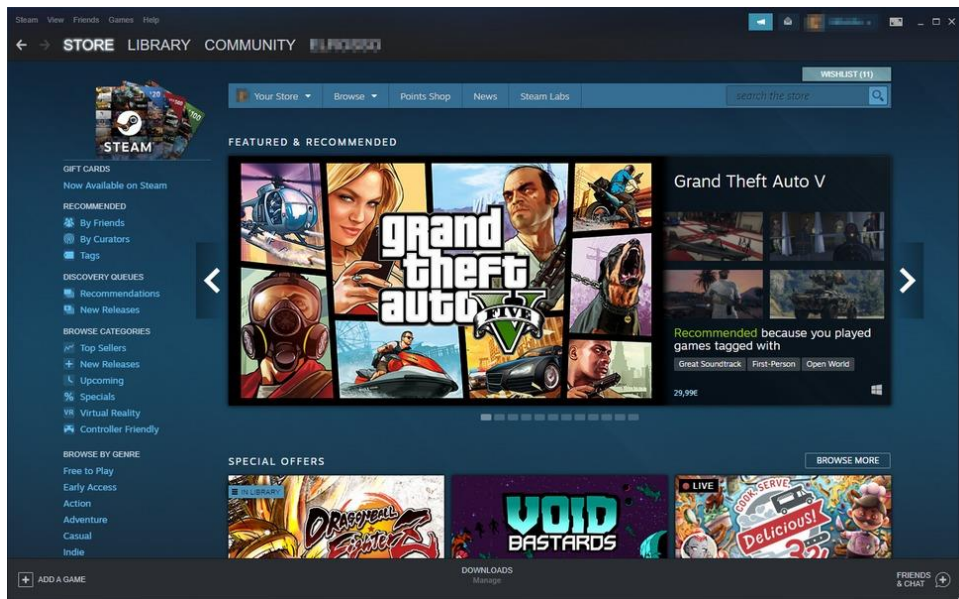
- Ο χρήστης να μπορεί να δει πολλές επιλογές σε παιχνίδια μαζί με τις λεπτομέρειες τους.
- Να μπορεί να φιλτράρει τις κατηγορίες που θα βλέπει.
- Να βλέπει τα παιχνίδια σε ένα όμορφο ατμοσφαιρικό περιβάλλον.
- Να διατηρεί μία επεξεργάσιμη σελίδα αγαπημένων όπου μπορεί εύκολα να προσθέσει και να αφαιρέσει συγκεκριμένους τίτλους αλλά και να έχει όλες τις πληροφορίες των παιχνιδιών μπροστά τους χωρίς να χρειάζεται να επιλέγει το κάθε παιχνίδι.

[Παρόμοιες Εφαρμογές]

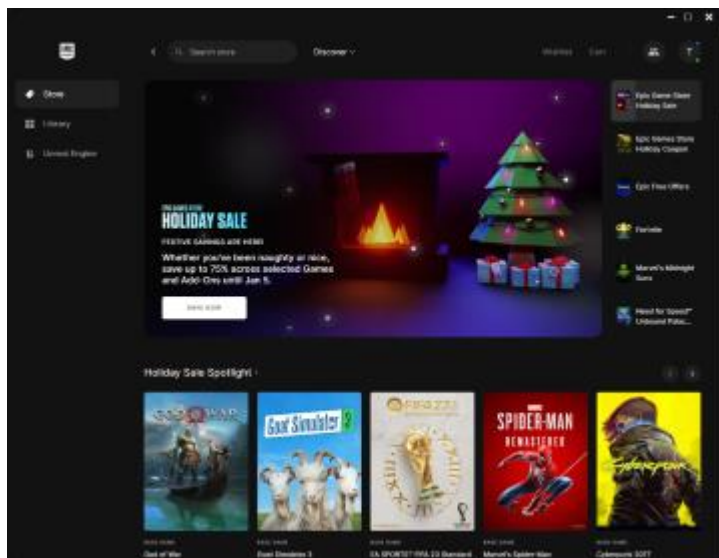
Γνωστές εφαρμογές που βοήθησαν στην ιδέα και υλοποίηση της παρούσας εφαρμογής είναι το Steam, το οποίο σήμερα φιλοξενεί πάνω από 120 εκατομμύριους μηνιαίους χρήστες και πάνω από 50 χιλιάδες παιχνίδια.



Το user interface του Steam είναι μινιμαλιστικό σε σχέδια και χρώματα και παρουσιάζει στον χρήστη μόνο την εικόνα του κάθε παιχνιδιού πριν εκείνος το επιλέξει.



Άλλη γνωστή εφαρμογή είναι το Epic Games Store, το οποίο ακολουθεί παρόμοια μινιμαλιστική σχεδίαση με το Steam και ένα σκουρόχρωμα θέμα χρωμάτων. Περισσότεροι από 31 εκατομμύρια χρήστες χρησιμοποιούν το Epic Games Store καθημερινά.



Η διαφοροποίηση της εφαρμογής βρίσκεται στο κομμάτι της εμφάνισης καθώς έχει ένα πιο ατμοσφαιρικό user interface με πιο έντονα χρώματα και πιο διαδραστικό, καθώς και στην εμφάνιση της πληροφορίας που δίνεται στον χρήστη καθώς δίνεται περισσότερη πληροφορία για το κάθε παιχνίδι πριν επιλεγεί από τον χρήστη.

[Περιγραφή εφαρμογής]

Η εφαρμογή αποτελείται από μια σελίδα Login όπου ο χρήστης εισάγει το email και το password του και έπειτα κατευθύνεται στην σελίδα όπου φαίνονται τα παιχνίδια.

Εδώ υπάρχουν κουμπιά για σύνδεση με official Gmail από την google το οποίο θα αναλυθεί παρακάτω, καθώς και σύνδεση με χρήση social platforms όπως το Instagram και το Whatsapp τα οποία δεν είναι λειτουργικά αυτήν την στιγμή. Όλα τα κουμπιά είναι διαδραστικά και κινούνται με τις κινήσεις του χρήστη στον κέρσορα.

Στην σελίδα των παιχνιδιών ο χρήστης μπορεί να επιλέξει την/τις κατηγορίες παιχνιδιών που επιθυμεί και η λίστα θα φιλτραριστεί ανάλογα με αυτές ώστε να προβάλλει μόνο τις κατηγορίες που θέλει. Η σελίδα έχει διαμορφωθεί με css ώστε να είναι διαδραστική και ευχάριστη για τον χρήστη, έτσι όταν ο χρήστης έχει τον κέρσορα πάνω από μια κάρτα παιχνιδιού αυτή “αιωρείται” και αποκτά ένα χρώμα background που δίνει την αίσθηση της αιώρησης καθώς και την κίνηση.

Επίσης σε αυτήν την σελίδα μπορεί να επιλέξει κάποιο παιχνίδι ώστε να το προσθέσει στη λίστα Αγαπημένων του στην οποία μπορεί να κατευθυνθεί με τον σύνδεσμο στο πάνω μέρος της σελίδας.

Στην σελίδα αγαπημένα(Favorites) μπορεί να δει τα επιλεγμένα παιχνίδια που έχει προσθέσει καθώς και να τα αφαιρέσει με το αντίστοιχο κουμπί, ενώ όλες οι πληροφορίες για το κάθε παιχνίδι παραμένουν ορατές .

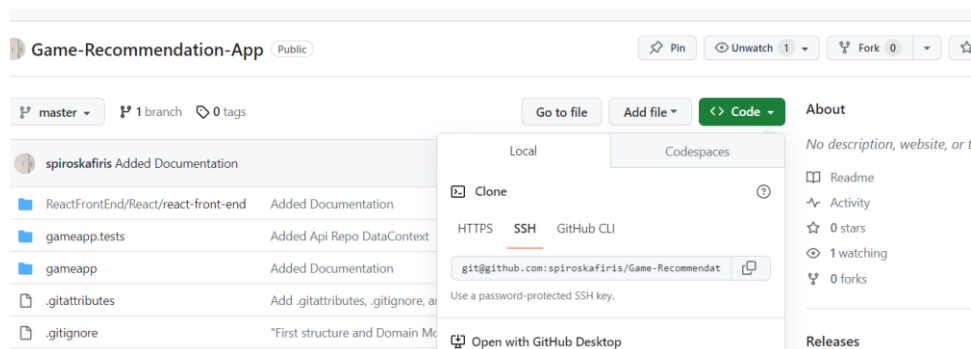
Όλες οι αλλαγές που γίνονται από τον χρήστη αποθηκεύονται στην βάση δεδομένων και παραμένουν την επόμενη φορά που θα ανοίξει την εφαρμογή.

[Ανάλυση απαιτήσεων]

Απαιτούνται εγκατεστημένα στον υπολογιστή το Visual Studio για το back-end της εφαρμογής, το Tableplus και λογαριασμό Elephantsql για φιλοξενία της βάσης δεδομένων στον Server, Visual Studio Code για τον κώδικα του front-end και , τέλος, σύνδεση στο διαδίκτυο. Σε περίπτωση που ο ενδιαφερόμενος κατεβάσει την εφαρμογή από το GitHub θα χρειαστεί και το Git Bash για να τρέξει την εντολή clone στον υπολογιστή του.

[Οδηγός Εγκατάστασης]

Όποιος επιθυμεί να χρησιμοποιήσει την εφαρμογή θα πρέπει να ακολουθήσει τις παρακάτω οδηγίες καθώς και να εγκαταστήσει τα παραπάνω προγράμματα.



Από το github μου:

<https://github.com/spiroskafiris/Game-Recommendation-App>

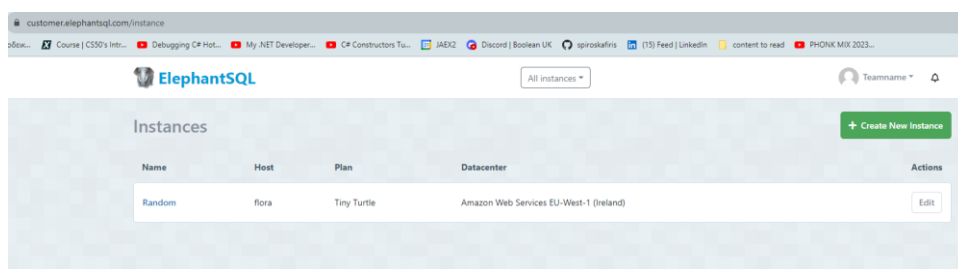
Θα πρέπει να επιλέξει το Code (πράσινο κουμπί) και έπειτα να αντιγράψει το κλειδί SSH που φαίνεται στην εικόνα.

Έπειτα θα πρέπει να ανοίξει ένα παράθυρο Git Bash στον υπολογιστή του στον φάκελο όπου θέλει να κατεβάσει(αντιγράψει/clone) την εφαρμογή με την εντολή “git clone” και το κλειδί SSH που αντέγραψε. (θα έχει τη μορφή “git clone [git@github.com:spiroskafiris/Game-Recommendation-App.git](https://github.com/spiroskafiris/Game-Recommendation-App.git)”)

Όνομα	Ημερομηνία τροποποι...	Τύπος	Μέγεθος
.github	16/10/2023 8:16 μμ	Φάκελος αρχείων	
gameapp	30/10/2023 4:17 μμ	Φάκελος αρχείων	
gameapp.tests	14/10/2023 11:34 μμ	Φάκελος αρχείων	
ReactFrontEnd	19/10/2023 7:19 μμ	Φάκελος αρχείων	
.gitattributes	14/10/2023 11:39 μμ	Έγγραφο κειμένου	3 KB
.gitignore	14/10/2023 11:43 μμ	Έγγραφο κειμένου	7 KB
Gameapp Intro	30/10/2023 7:22 μμ	Αρχείο PNG	38 KB
gameapp.sln	15/10/2023 12:09 πμ	Visual Studio Solu...	2 KB
GameAppDoc	3/11/2023 8:29 μμ	Microsoft Word D...	8.397 KB
GameAppDocumentation	3/11/2023 8:30 μμ	Microsoft Edge P...	1.752 KB
gameappmodel	15/10/2023 12:08 πμ	Αρχείο JPG	71 KB
games	31/10/2023 3:44 μμ	Αρχείο PNG	1.897 KB
home	31/10/2023 3:44 μμ	Αρχείο PNG	1.366 KB
Layout	31/10/2023 3:49 μμ	Αρχείο PNG	1.751 KB
login	31/10/2023 3:44 μμ	Αρχείο PNG	1.643 KB
README	14/10/2023 11:39 μμ	Markdown Source...	1 KB


Ο φάκελος που θα δημιουργηθεί θα είναι της μορφής της εικόνας.

Ο χρήστης έπειτα θα χρειαστεί να δημιουργήσει έναν λογαριασμό στην ιστοσελίδα <https://www.elephantsql.com/> και θα μεταφερθεί στην παρακάτω σελίδα.








Εδώ πρέπει να επιλέξει το Create New Instance και να κρατήσει τα παρακάτω στοιχεία.

(Τα στοιχεία διαφέρουν για τον κάθε χρήστη)

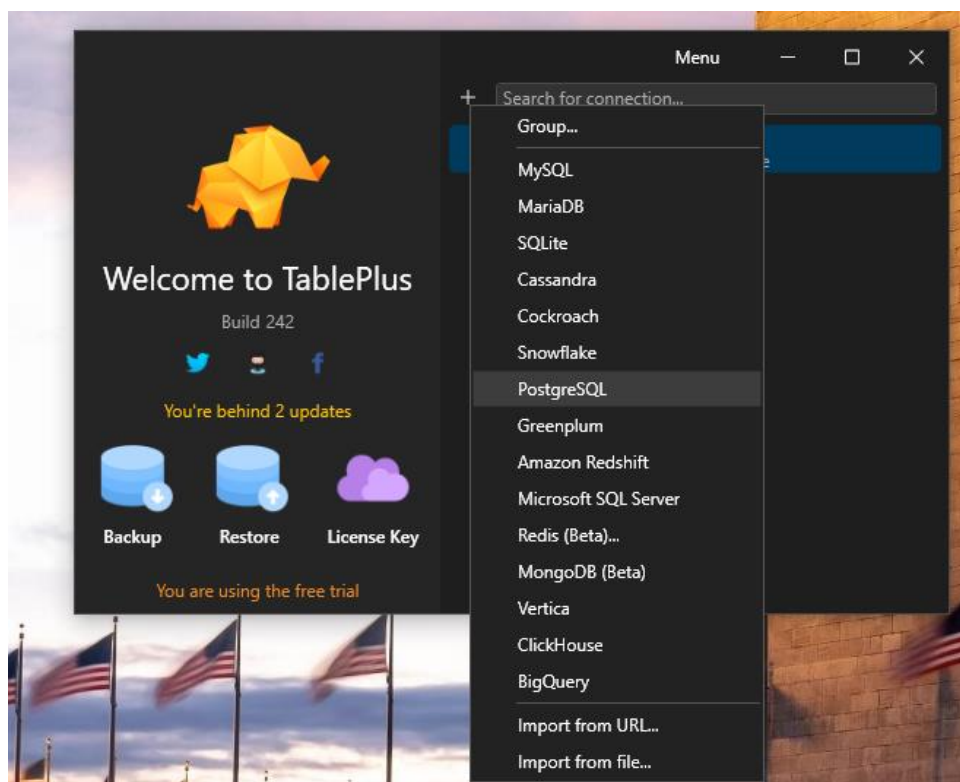
 ElephantSQL

Random

Details

Server	flora.db.elephantsql.com (flora-01)	
Region	amazon-web-services:eu-west-1	
Created at	2023-10-07 14:33 UTC+00:00	
User & Default database	krllrmxe	Reset
Password	***  	 Rotate password
URL	postgres://krllrmxe:***@flora.db.elephantsql.com/krllrmxe  	
Current database size	512 KB	
Max database size	20 MB	

Έπειτα, αφού ο χρήστης κατεβάσει το Tableplus από το <https://tableplus.com/download> για το δικό του σύστημα θα πρέπει να επιλέξει το + όπως φαίνεται στην εικόνα και την postgresql.



Και ακολούθως να συμπληρώσει τα στοιχεία του από τον elephantsql στα κατάλληλα πεδία όπως φαίνεται στην επόμενη εικόνα

PostgreSQL

Name

Status color Tag

Host Port

User Other options

Password Store in keychain

Database Bootstrap commands...

SSL mode

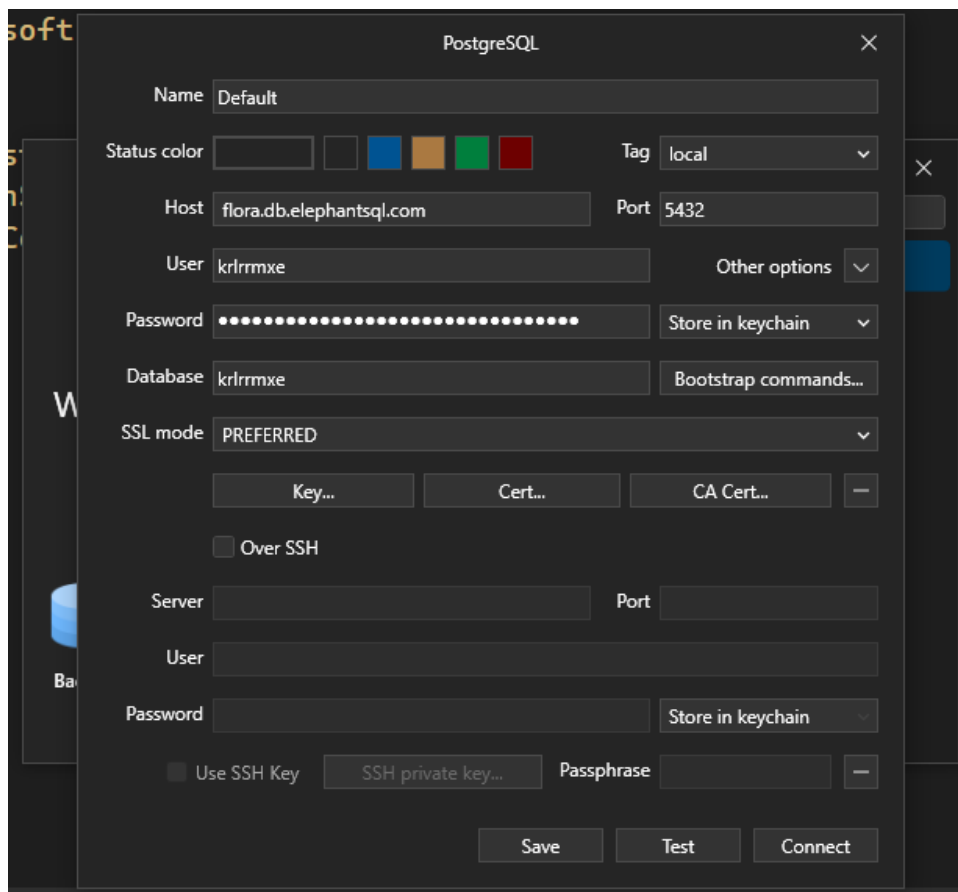
☐ Over SSH

Server Port

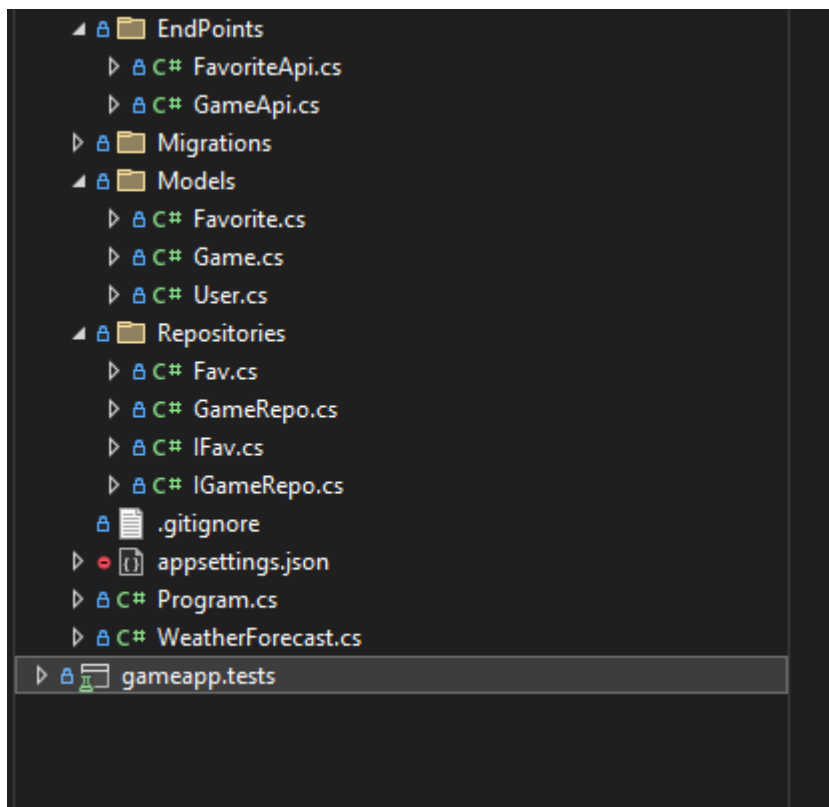
User

Password Store in keychain

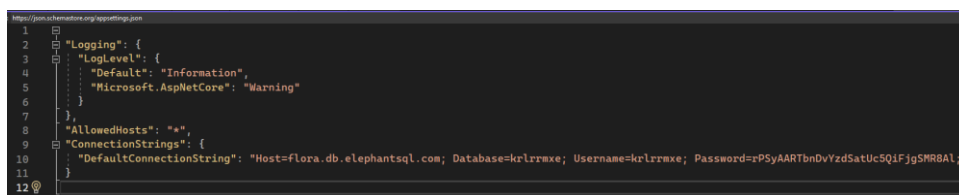
☐ Use SSH Key Passphrase



Από εδώ με το Visual Studio θα πρέπει να ανοίξει το gameapp.sln το οποίο θα ανοίξει τον κώδικα για το back end.

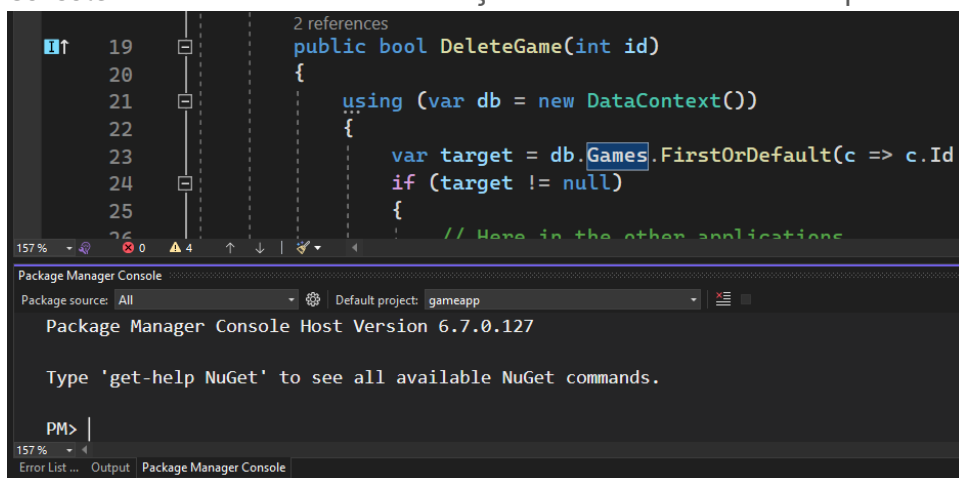


Εδώ πρέπει να επιλέξει το appsettings.json.



Εδώ πρέπει να αντιγράψει τα δικά του στοιχεία από τον ElephantSql όπως φαίνονται στην παραπάνω εικόνα.

Μέσα στο Visual Studio επιλέγοντας Tools>NuGet Package Manager> Package Manager Console θα ανοίξει το παρακάτω εργαλείο.



Εδώ πρέπει να εκτελεστούν οι εξής εντολές:

```
add-migration FirstMigration
```

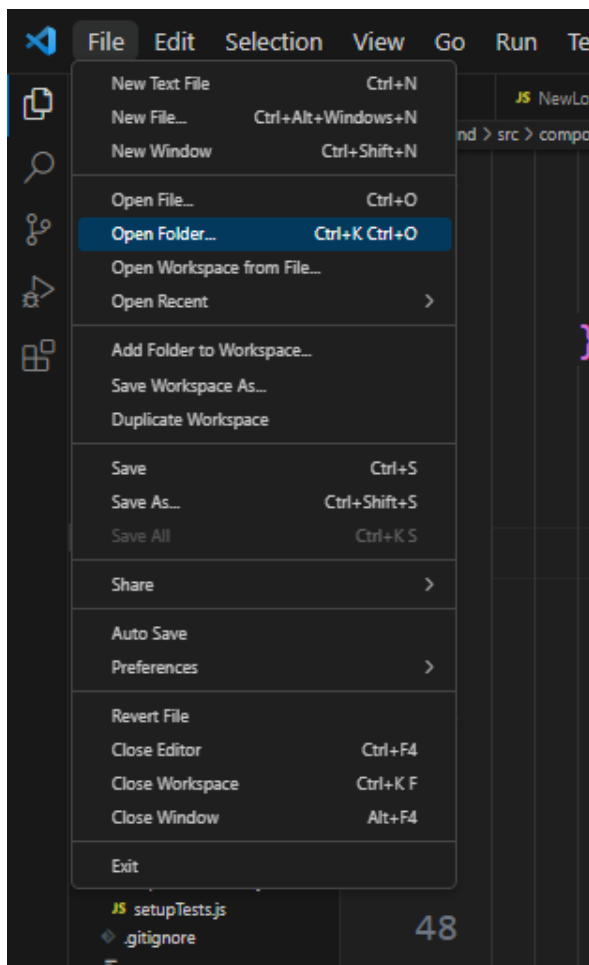
```
update-database
```

Οι οποίες εντολές θα δημιουργήσουν την βάση δεδομένων στο tableplus.

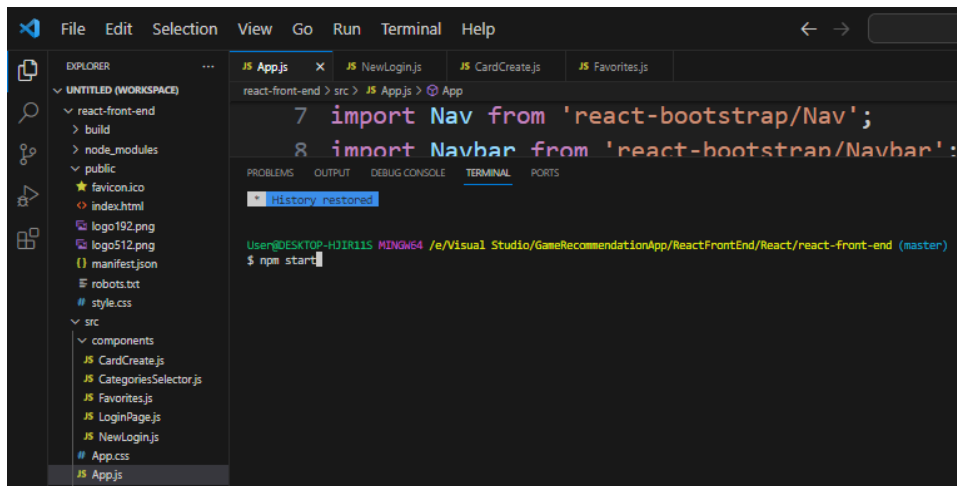
Αφού ολοκληρωθούν τα παραπάνω βήματα ο χρήστης μπορεί να πατήσει F5 στο Visual Studio ώστε να εκτελεστεί η εφαρμογή και να ανοίξει το SwaggerApi.

Επόμενο βήμα είναι το άνοιγμα του Visual Studio Code και του κώδικα του front end.

Ανοίγοντας το επιλέγουμε File> Open Folder και βρίσκουμε το ReactFrontEnd μέσα στο φάκελο που δημιουργήθηκε από το GitHub.



Έπειτα επιλέγουμε Terminal> New Terminal και στο τερματικό που ανοίγει εκτελούμε την εντολή npm start όπως φαίνεται στην εικόνα



Η εφαρμογή θα πρέπει να ανοίξει μετά από μερικά λεπτά.

[Σχεδιασμός]

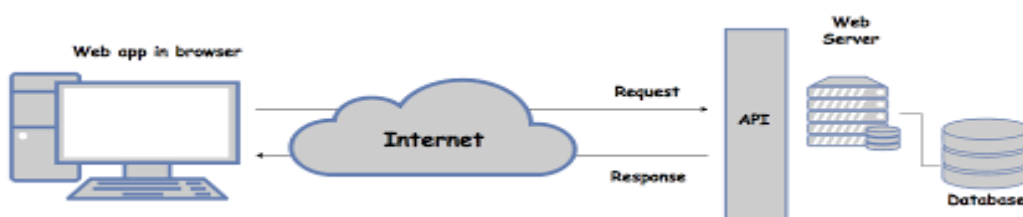
Η παρουσίαση της αρχιτεκτονικής θα γίνει από το back-end προς το front-end και την τελική εφαρμογή αλλά αν είναι επιθυμητό μπορείτε να γίνει και αντίστροφα.

Η εφαρμογή είναι σχεδιασμένη για προσωπική χρήση (ένας χρήστης) αν και υπάρχει κώδικας για υποστήριξη περισσότερων χρηστών που θα ολοκληρωθεί στο μέλλον.

[Τεχνολογίες που χρησιμοποιήθηκαν]

Visual Studio: Το επέλεξα λόγω της ενασχόλησης με αυτό κατά το χρόνια φοίτησής μου στο Πανεπιστήμιο Πειραιώς και λόγω προσωπικής αρεσκείας και άνεσης.

MVC: Ακολούθησα την αρχιτεκτονική MVC για το back end της εφαρμογής. Το Model-View-Controller είναι ένα μοντέλο αρχιτεκτονικής λογισμικού το οποίο χρησιμοποιείται για τη δημιουργία περιβαλλόντων αλληλεπίδρασης χρήστη. Στο μοντέλο αυτό η εφαρμογή διαιρείται σε τρία διασυνδεδεμένα μέρη ώστε να διαχωριστεί η παρουσίαση της πληροφορίας στον χρήστη από την μορφή που έχει αποθηκευτεί στο σύστημα.



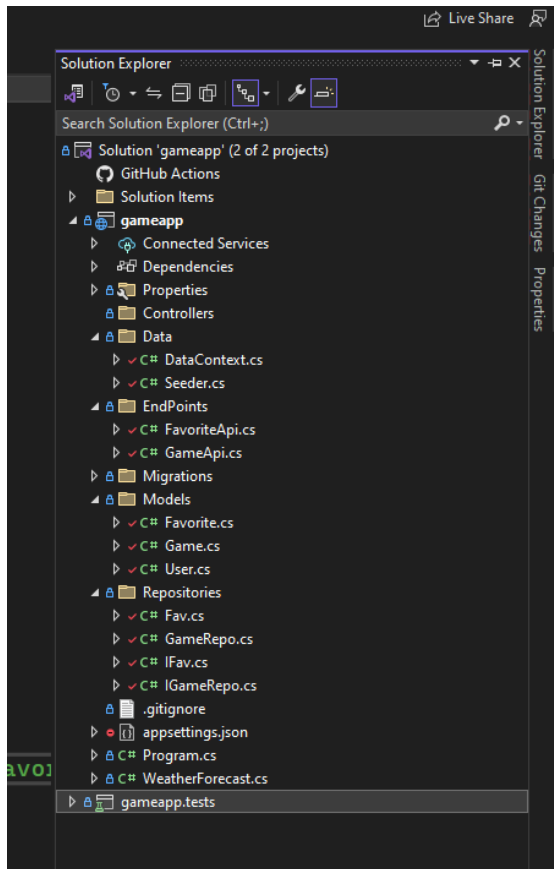
Tableplus: Χρησιμοποιείται για την διαχείριση της βάσης δεδομένων η οποία είναι σε ElephantSql. Είναι ευκολόχρηστο και μικρό σε μέγεθος.

Visual Studio Code: Για τον κώδικα του front-end σε Html/Javascript/React λόγω ευκολίας στη χρήση και του ότι είναι αρκετά διαδεδομένο.

Όλα τα προγράμματα είναι δωρεάν για χρήση.

{Back-end}

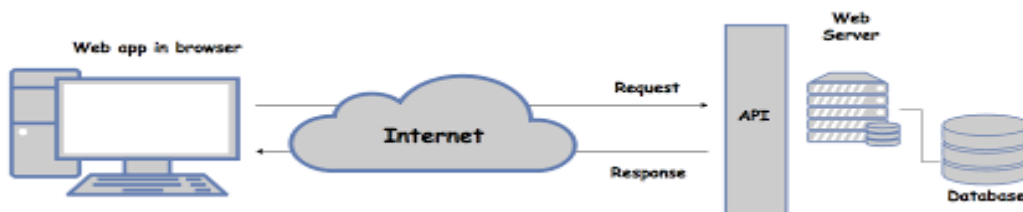
Η δομή του κώδικα σε C# με χρήση Visual Studio είναι η παρακάτω.



Data/DataContext: Εδώ γίνεται η σύνδεση με τη βάση δεδομένων και η αρχικοποίηση της βάσης Games όπου αποθηκεύονται τα παιχνίδια.

Data/Seeder: Εδώ εισάγουμε τα παιχνίδια στην βάση όταν αυτή είναι κενή.

Endpoints/ GameApi :



Η δομή του GameApi αποτελείται από το διάγραμμα της φωτογραφίας. Η βάση περιέχει τα παιχνίδια (αφού εισαχθούν από την κλάση Seeder).

Ο elephantsql server φιλοξενεί την βάση και το GameApi κάνει τις διάφορες λειτουργίες πάνω στην βάση, όπως Get(Εξαγωγή) Put(Επεξεργασία) Post(Εισαγωγή) Delete(Διαγραφή) και άλλες.

Αφού το GameApi κάνει κάποια λειτουργία αυτή περνάει στο front-end της εφαρμογής και εμφανίζεται στον χρήστη.

```
//get
[HttpGet]
[ProducesResponseType(StatusCodes.Status200OK)]
1 reference
private static async Task<IResult> GetAllGames(IGameRepo service)
{
    try
    {
        return await Task.Run(() => {
            return Results.Ok(service.GetAllGames());
        });
    }
    catch (Exception ex)
    {
        return Results.Problem(ex.Message);
    }
}
```

Εδώ, μέσα στο GameApi, υπάρχουν όλες οι λειτουργίες όπως η GetAllGames όπου ο χρήστης θα ‘πάρει’ όλα τα παιχνίδια από την βάση για να τα δει.

Models/Game: Εδώ βρίσκεται η κλάση Game που θα αποτελέσει το κάθε παιχνίδι καθώς και οι κλάσεις Favorite και User που αποτελούν μελλοντικές ιδέες/προσθήκες και στην παρούσα φάση δεν χρησιμοποιούνται.

```

namespace gameapp.Models
{
    64 references
    public class Game
    {
        6 references
        public int Id { get; set; }
        27 references
        public string Title { get; set; }
        25 references
        public string Img { get; set; }
        27 references
        public string Developer { get; set; }
        27 references
        public List<string> Genres { get; set; }
        27 references
        public int Rating { get; set; }
        27 references
        public string Description { get; set; }
        29 references
        public bool isFavorite { get; set; } //this
    }
}

```

Id: Το primary key του παιχνιδιού που είναι μοναδικό για το κάθε παιχνίδι.

Title: Ο τίτλος του παιχνιδιού.

Img: Το url της εικόνας του παιχνιδιού.

Developer: Ο σχεδιαστής του παιχνιδιού.

Genres: Η λίστα με τις διάφορες κατηγορίες που ανήκει το κάθε παιχνίδι.

Rating: Η βαθμολογία του παιχνιδιού.

Description: Σύντομη περιγραφή του παιχνιδιού.

IsFavorite: μεταβλητή αληθούς/ψευδούς για το αν το παιχνίδι είναι στα αγαπημένα του χρήστη ή όχι.

Repositories/IGameRepo: Εδώ αρχικοποιούνται οι λειτουργίες που θα μπορούν να γίνουν στα παιχνίδια.

```

namespace gameapp.Repositories
{
    9 references
    public interface IGameRepo
    {
        2 references
        bool AddGame(Game game);
        2 references
        Game GetGame(int id);
        2 references
        IEnumerable<Game> GetAllGames();
        2 references
        bool DeleteGame(int id);
        2 references
        bool UpdateGame(Game game);
        2 references
        bool changeIsFavtoTrue(int id);
        2 references
        bool changeIsFavtoFalse(int id);
    }
}

```

Όλες οι λειτουργίες είναι υποχρεωτικό να αναπτυχθούν και αυτό γιατί το IGameRepo αποτελεί Interface για την κλάση GameRepo που θα ακολουθήσει.

AddGame: προσθήκη παιχνιδιού

GetGame: επιλογή συγκεκριμένου παιχνιδιού με βάση το Id.

GetAllGames: επιλογή όλων των παιχνιδιών στην βάση.

DeleteGame: διαγραφή παιχνιδιού με βάση το Id.

UpdateGame: επεξεργασία των ιδιοτήτων του παιχνιδιού, οποιαδήποτε πληροφορία μπορεί να αλλαχθεί και αποθηκευτεί εκ νέου.

ChangelsFavtoTrue/changelsFavtoFalse: αλλαγή της μεταβλητής isFavorite σε true/false αντίστοιχα.

Repositories/GameRepo: Ανάπτυξη των λειτουργιών που προαναφέρθηκαν.

```

2 references
public IEnumerable<Game> GetAllGames()
{
    using (var db = new DataContext())
    {
        return db.Games.ToList();
    }
    return null;
}

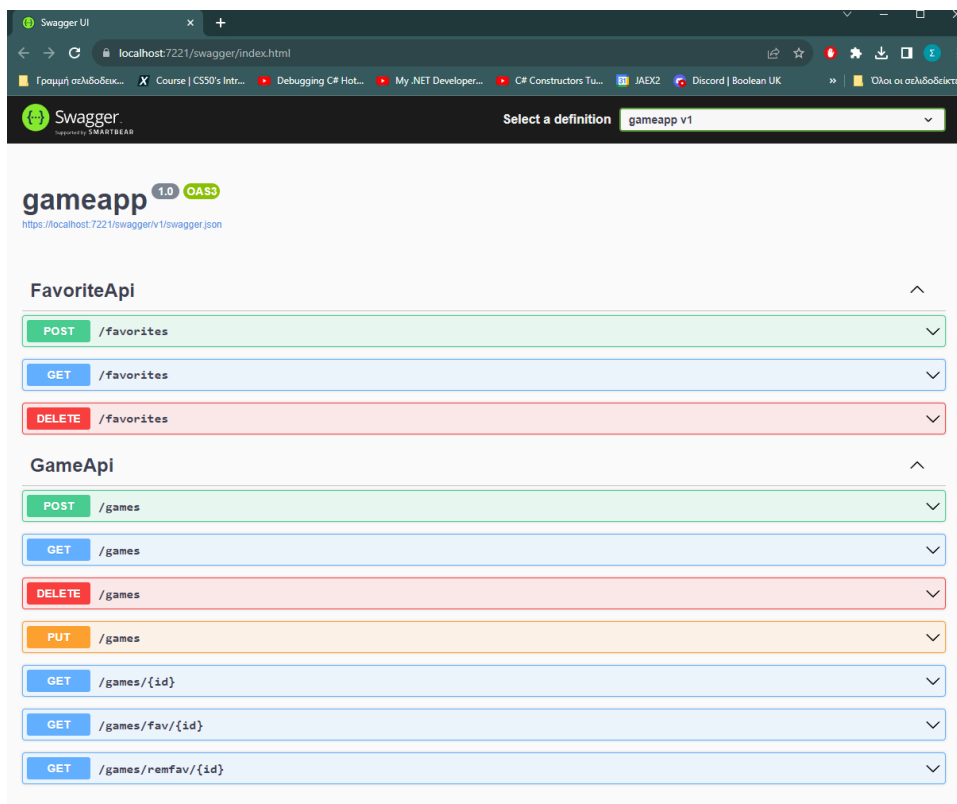
```

Για την GetAllGames χρησιμοποιούμε μια μεταβλητή βάσης db για να χρησιμοποιήσουμε την βάση Games που έχουμε αρχικοποιήσει στο DataContext και να μετατρέψουμε όλη την βάση σε μια λίστα ώστε να την προβάλουμε στον χρήστη.

Ομοίως αναπτύσσονται και οι υπόλοιπες λειτουργίες.

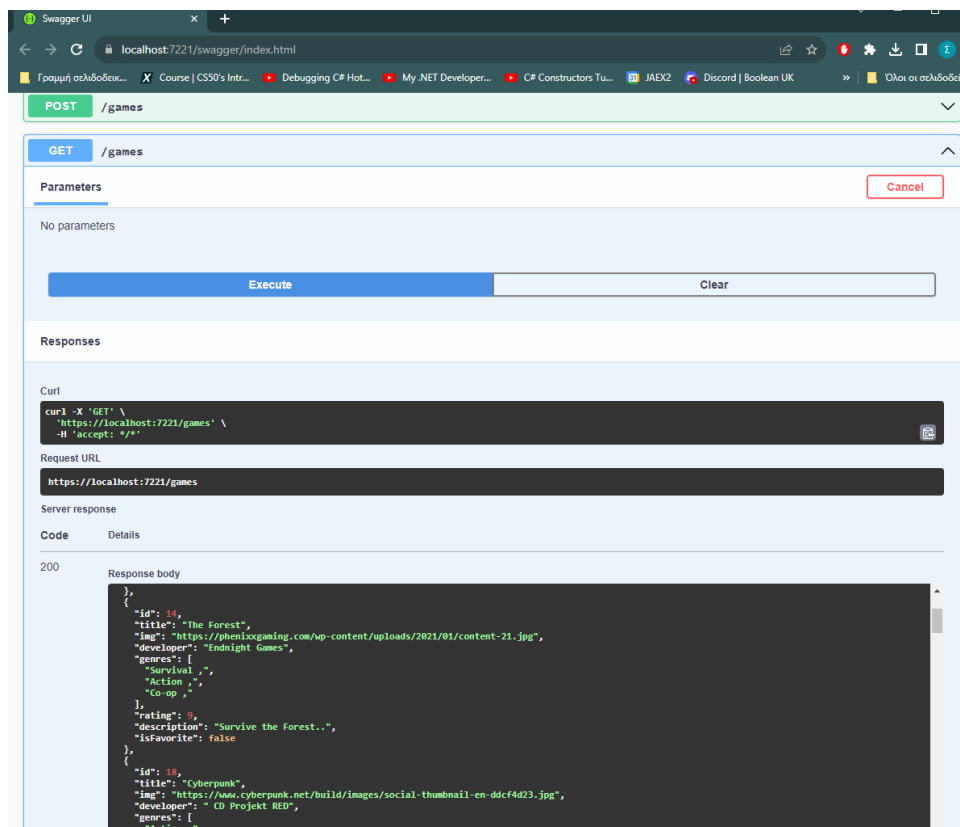
Program.cs: Κλάση αυτόματα δημιουργημένη από την βιβλιοθήκη dotnet όπου προστέθηκαν κάποιες γραμμές κώδικα απαραίτητες για τις παραπάνω λειτουργίες.

Το SwaggerApi που δημιουργήθηκε για την εφαρμογή είναι το παρακάτω.



Το GameApi κάνει όλες τις λειτουργίες που προαναφέραμε.

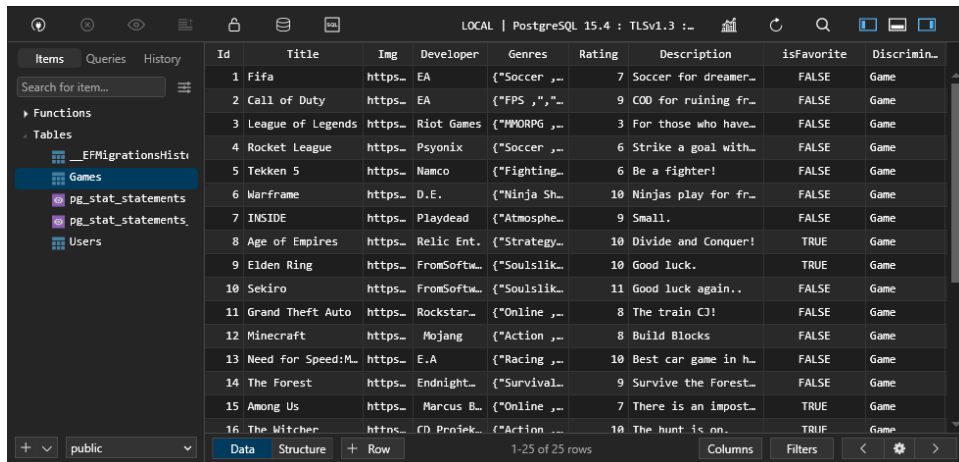
Παραδείγματος χάρι η Get /games που δείχνει στον χρήστη όλα τα παιχνίδια της βάσης.



Το FavoriteApi καθώς και τα Favorite, User, IFav, Fav αποτελούν διαφορετικό τρόπο εφαρμογής της λειτουργίας Αγαπημένων που θα πραγματοποιηθεί στο μέλλον.

{Βάση Δεδομένων- TablePlus}

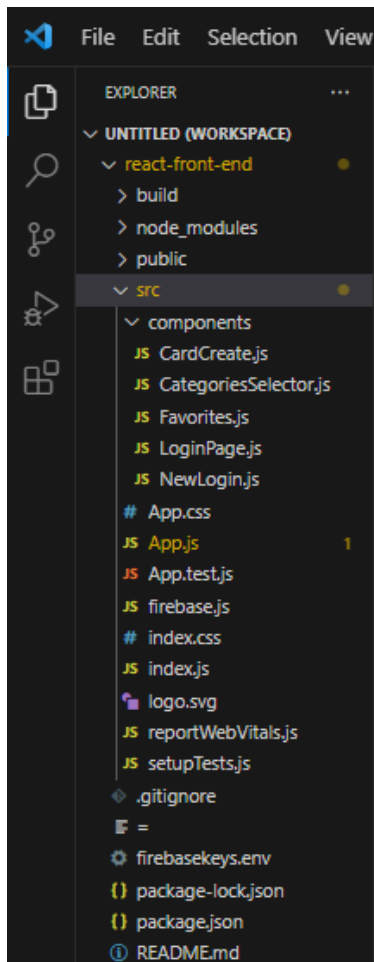
Η Βάση και ο πίνακας Games με όλα τα δεδομένα.



Id	Title	Img	Developer	Genres	Rating	Description	isFavorite	Discrimin...
1	Fifa	https...	EA	{ "Soccer", ...	7	Soccer for dreamer...	FALSE	Game
2	Call of Duty	https...	EA	{ "FPS", "...	9	COD for ruining fr...	FALSE	Game
3	League of Legends	https...	Riot Games	{ "MMORPG", ...	3	For those who have...	FALSE	Game
4	Rocket League	https...	Psyonix	{ "Soccer", ...	6	Strike a goal with...	FALSE	Game
5	Tekken 5	https...	Namco	{ "Fighting", ...	6	Be a fighter!	FALSE	Game
6	Warframe	https...	D.E.	{ "Ninja Sh...	10	Ninjas play for fr...	FALSE	Game
7	INSIDE	https...	Playdead	{ "Atmosphe...	9	Small.	FALSE	Game
8	Age of Empires	https...	Relic Ent.	{ "Strategy", ...	10	Divide and Conquer!	TRUE	Game
9	Elden Ring	https...	FromSoftw...	{ "Soulslik...	10	Good luck.	TRUE	Game
10	Sekiro	https...	FromSoftw...	{ "Soulslik...	11	Good luck again..	FALSE	Game
11	Grand Theft Auto	https...	Rockstar...	{ "Online", ...	8	The train CJ!	FALSE	Game
12	Minecraft	https...	Mojang	{ "Action", ...	8	Build Blocks	FALSE	Game
13	Need for Speed: M...	https...	E.A	{ "Racing", ...	10	Best car game in h...	FALSE	Game
14	The Forest	https...	Endnight...	{ "Survival", ...	9	Survive the Forest...	FALSE	Game
15	Among Us	https...	Marcus B...	{ "Online", ...	7	There is an impost...	TRUE	Game
16	The Witcher	https...	CD Projekt...	{ "Action", ...	10	The hunt is on.	TRUE	Game

{Front-end}

Ακολουθεί η παρουσίαση του κώδικα για το μέρος της εφαρμογής που προβάλλεται στον χρήστη με χρήση Visual Studio Code.



Ο κώδικας που χρησιμοποιείται βρίσκεται στα CardCreate, Favorites, NewLogin, App.js, index.js καθώς και το style.css μέσα στο public.

CardCreate:

```

10 function App() {
11   return(
12     <>
13     <header>
14       <h1>Game Recommendations App</h1>
15       <Navbar className="navbar">
16         <Container>
17           <Nav className="nav-elements">
18             <Nav.Link href="/NewLogin" className="nav-elements">Log in</Nav.Link>
19             <Nav.Link href="/CardCreate" className="nav-elements">Categories</Nav.Link>
20             <Nav.Link href="/Favorites" className="nav-elements">Favorites</Nav.Link>
21           </Nav>
22         </Container>
23       </Navbar>
24       <nav>
25         <ul>
26           <Routes>
27             <Route path="/NewLogin" element={<NewLogin/>} />
28             <Route path="/CardCreate" element={<CardCreate/>} />
29             <Route path="/Favorites" element={<Favorites/>} />
30           </Routes>
31         </ul>
32       </nav>
33     </header>
34   </>)
35 }
36
37 export default App;

```

Σειρές 15-23: Μέρος της σελίδας για κατεύθυνση του χρήστη στα εκάστοτε μέρη της εφαρμογής.

Σειρές 26-30: Κατευθύνσεις για τα διάφορα link μέσα στην εφαρμογή.

NewLogin:

```

12   return(
13     <section>
14       <div className="image">
15         
16       </div>
17
18       <div className="content">
19         <div className="form">
20           <h3>Login</h3>
21           <form action="#">
22             <div className="input">
23               <span>Username</span>
24               <input type="text" name="username" required/>
25             </div>
26             <div className="input">
27               <span>Password</span>
28               <input type="password" name="password" required/>
29             </div>
30             <div className="input">
31               <input type="submit" value="Submit" id="submit-button" onClick={handleSubmit}/>
32             </div>
33             <div className="input">
34               <p>Don't have an account? <a href="#">Sign up</a></p>
35             </div>
36             <div className="input">
37               <button id="google-login-btn" className="google-button"><i className="fab fa-google"></i>
button
38             </div>
39           </form>
40         </div>

```

Ο κώδικας για την σελίδα Login αποτελείται από μια φόρμα όπου ο χρήστης εισάγει τα στοιχεία του (email,password) καθώς και κάποια κουμπιά για διαδικασίες που πρόκειται να υλοποιηθούν στο μέλλον. Το σχόλιο είναι μέρος λειτουργίας που επιτρέπει στον χρήστη να

συνδεθεί χρησιμοποιώντας το Google email του μέσω της επίσημης εφαρμογής και αποτελεί μελλοντική προσθήκη.

```
41      <h3>Login with Social media</h3>
42      <ul className="social-media">
43        <li></img></li>
45        <li></img></li>
47        <li></img></li>
49      </ul>
    </div>
  </div>
</section>
```

Στην σελίδα ο χρήστης μπορεί επίσης να δει επιλογές σύνδεσης με άλλα μέσα κοινωνικής δικτύωσης που θα προστεθούν στο μέλλον.

CardCreate: εδώ υπάρχει η πλειονότητα των λειτουργιών και η κλήση του back-end.

```
<h1>The best games for you are:</h1>
<Row xs={1} md={3} className="cards">
  {
    filteredDATA.map(filteredGames => (
      <Col key={filteredGames.id}>
        <Card>
          <Card.Img
            className="card--img" variant="top" src={filteredGames.img} />
          <Card.Body className="card--text">
            <Card.Title><b>Title:</b> {filteredGames.title}</Card.Title>
            <Card.Text>
              <b>Developer:</b> {filteredGames.developer}
            </Card.Text>
            <Card.Text>
              <b>Genres:</b> {filteredGames.genres}
            </Card.Text>
            <Card.Text>
              <b>Rating:</b> {filteredGames.rating}
            </Card.Text>
            <Card.Text>
              <b>Description:</b> {filteredGames.description}
            </Card.Text>
            <button id="addtofav-btn" className="addtofav-btn" onClick={(e) => addtofav(filteredGames.id)}>Favorite</button>
          </Card.Body>
        </Card>
      </Col>
    ))
  }
</Row>
```

Εδώ δημιουργούνται οι ‘κάρτες’ του κάθε παιχνιδιού με χρήση map σε φιλτραρισμένα παιχνίδια οπότε ο χρήστης έχει επιλέξει κάποια κατηγορία ή καμία, οπότε εμφανίζονται όλα τα παιχνίδια.

```

<>
<h1>Choose a Game Category:</h1>

<label className="container2">Action
  <input type="checkbox" value="Action ," onChange={handleChange}/>
  <span className="checkmark"></span>
</label>
<label className="container2">Co-op
  <input type="checkbox" value="Co-op ," onChange={handleChange}/>
  <span className="checkmark"></span>
</label>
<label className="container2">Online
  <input type="checkbox" value="Online ," onChange={handleChange}/>
  <span className="checkmark"></span>
</label>
<label className="container2">Fighting
  <input type="checkbox" value="Fighting ," onChange={handleChange}/>
  <span className="checkmark"></span>
</label>
<label className="container2">FPS
  <input type="checkbox" value="FPS ," onChange={handleChange}/>
  <span className="checkmark"></span>
</label>
<label className="container2">Cars
  <input type="checkbox" value="Cars ," onChange={handleChange}/>
  <span className="checkmark"></span>
</label>
<label className="container2">Ninja Shooter
  <input type="checkbox" value="Ninja Shooter ," onChange={handleChange}/>
  <span className="checkmark"></span>
</label>

```

Εδώ δημιουργείται το μέρος της σελίδας που επιλέγονται κατηγορίες παιχνιδιών ώστε να φιλτραριστεί η προβολή των παιχνιδιών.

```

8 function CardCreate() {
9   const [gameData, setGameData] = useState([]);
10  const [filterTags, setFilterTags] = useState([])
11
12  useEffect(() => {
13    axios.get("https://localhost:7221/games").then((response) => {
14      //console.log(response.data)
15      setGameData((existingData) => {
16        return response.data;
17      });
18    });
19  }, []);
20
21  const filteredDATA = gameData.filter((game) =>
22  filterTags.length > 0
23    ? filterTags.every((filterTag) =>
24      game.genres.includes(filterTag)
25    )
26    : gameData
27  )
28
29  const handleChange = (event) => {
30    if (event.target.checked) {
31      setFilterTags([...filterTags, event.target.value])
32    } else {
33      setFilterTags(
34        filterTags.filter((filterTag) => filterTag !== event.target.value)
35      )
36    }
37  }
38

```

gameData: εδώ αποθηκεύεται η βάση Games στο front-end.

filterTags: τα φίλτρα που επιλέγονται/αποεπιλέγονται από τον χρήστη.

Σειρές 12-19: κλήση του GameApi και της λειτουργίας GetAllGames από το back-end και αποθήκευση των δεδομένων στο gameData.

Σειρές 21-27: φιλτράρισμα του gameData πριν σταλεί στο map με τη μορφή filteredDATA και μορφοποιηθεί σε 'карτες'.

Σειρές 29-37: προσθήκη κατηγορίας παιχνιδιών όταν επιλεγεί από τον χρήστη, αφαίρεση όταν αποεπιλεγεί.

```

39     function addtofav(id) {
40         // var payload = {
41         //     id:id,
42         //     img: gameData[id-1].img,
43         //     title: gameData[id-1].title,
44         //     developer: gameData[id-1].developer,
45         //     genres: gameData[id-1].genres,
46         //     rating: gameData[id-1].rating,
47         //     description: gameData[id-1].description,
48         //     isFavorite: true
49         // }
50         // console.log(payload)
51
52         axios.get('https://localhost:7221/games/fav/' + id)
53             .then((response) => {
54                 console.log(response.data);
55             })
56             .catch((error) => {
57                 console.error(error);
58                 console.log("failed");
59             });
60     }

```

addtofav: Εδώ γίνεται η αλλαγή της μεταβλητής isFavorite του κάθε παιχνιδιού με χρήση get όπου καλείται η αντίστοιχη μέθοδος από τον GameApi του back-end μαζί με το id του συγκεκριμένου παιχνιδιού. Καλείται με το κουμπί Favorite πάνω σε κάθε 'κάρτα'.

Τα σχόλια αποτελούν κώδικα που θα χρησιμοποιούταν με διαφορετική μέθοδο για την προσθήκη Αγαπημένων.

Favorites: ο κώδικας της σελίδας όπου ο χρήστης βλέπει τα Αγαπημένα παιχνίδια του και αφαιρεί από αυτή όποια επιθυμεί.


```

6 import Row from 'react-bootstrap/Row';
7
8 function Favorites() {
9   const [gameData, setGameData] = useState([]);
10
11   useEffect(() => {
12     axios.get("https://localhost:7221/games").then((response) => {
13       //console.log(response.data)
14       setGameData((existingData) => {
15         return response.data;
16       });
17     });
18   }, []);
19
20   function removefromfav(id){
21     // var payload = {
22     //   id:id,
23     //   img: gameData[id-1].img,
24     //   title: gameData[id-1].title,
25     //   developer: gameData[id-1].developer,
26     //   genres: gameData[id-1].genres,
27     //   rating: 50,
28     //   description: gameData[id-1].description,
29     //   isFavorite: false
30     // }
31     // console.log(payload)
32
33     axios.get('https://localhost:7221/games/remfav/' + id)
34       .then((response) => {
35         console.log(response.data);

```

Σειρές 11-18: ομοίως με σειρές 12-19 του CardCreate που προαναφέρθηκε.

Σειρές 20- : ομοίως με addtofav του CardCreate για αλλαγή του isFavorite σε false και αφαίρεση του παιχνιδιού από τα Favorites.

```

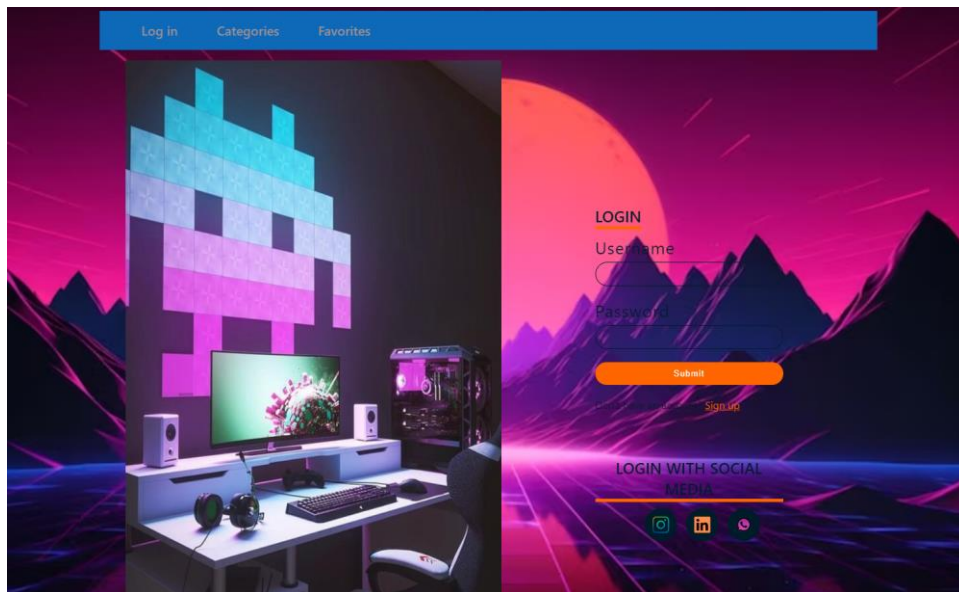
<h1>Your favorite games are:</h1>
<Row xs={1} md={3} className="cards">
  {
    gameData.filter((game) => game.isFavorite===true).map(filteredGames => (
      <Col key={filteredGames.id}>
        <Card>
          <Card.Img
            className="card--img" variant="top" src={filteredGames.img} />
          <Card.Body className="card--text">
            <Card.Title ><b>Title: </b>{filteredGames.title}</Card.Title>
            <Card.Text >
              <b>Developer:</b> {filteredGames.developer}
            </Card.Text>
            <Card.Text >
              <b>Genres:</b> {filteredGames.genres}
            </Card.Text>
            <Card.Text >
              <b>Rating:</b> {filteredGames.rating}
            </Card.Text>
            <Card.Text >
              <b>Description:</b> {filteredGames.description}
            </Card.Text>
            <button id="addtofav-btn" className="addtofav-btn" onClick={(e) =>
          </Card.Body>
        </Card>
      </Col>
    )
  )}
</Row>

```

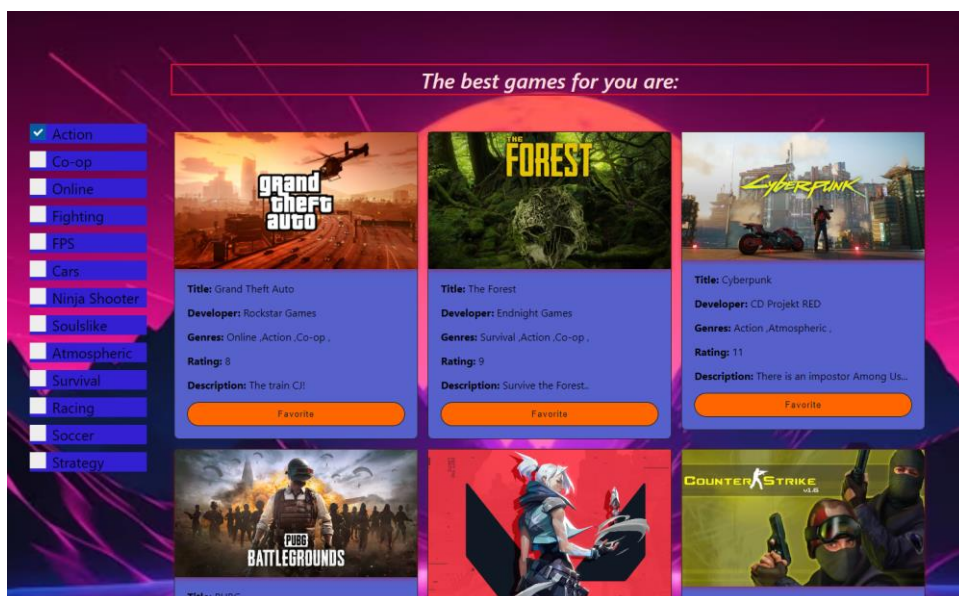
Δημιουργία ‘καρτών’ για προβολή των Αγαπημένων παιχνιδιών του χρήστη και κουμπί Remove για αφαίρεση αυτών.

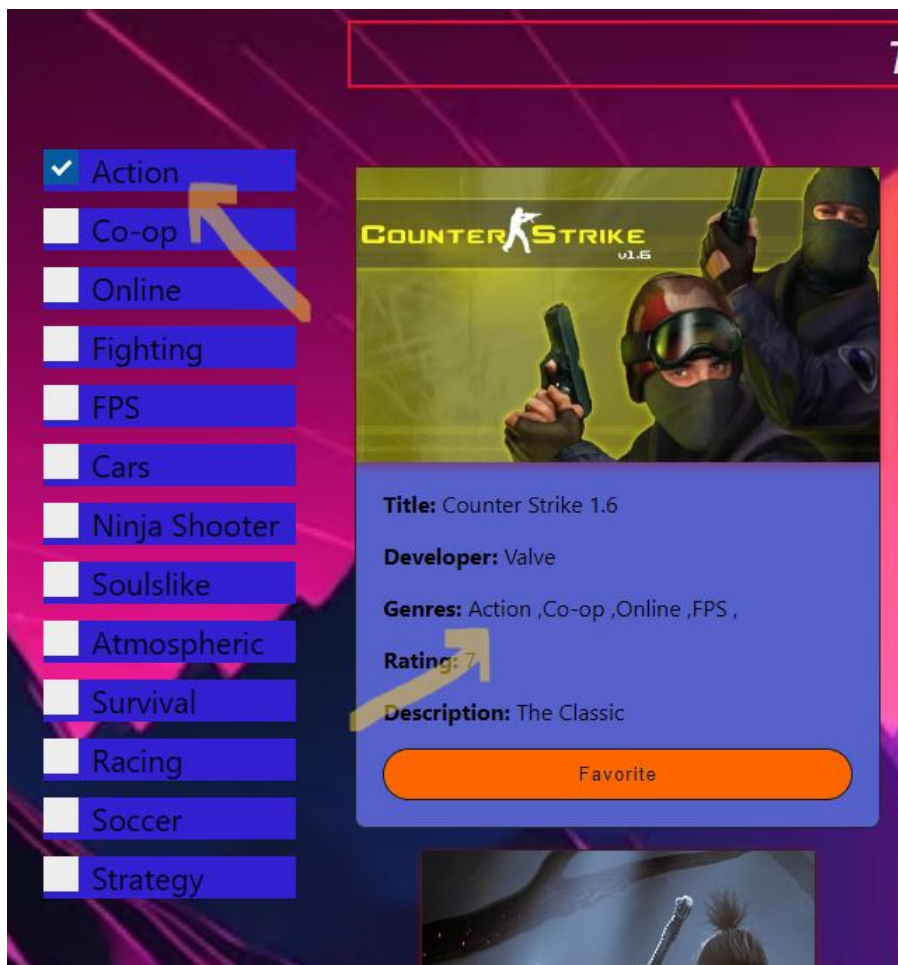
Στο style.css βρίσκονται όλες οι παράμετροι που χρησιμοποίησα για την μορφοποίηση της εφαρμογής.

[Web Game Showcase τελική εφαρμογή]



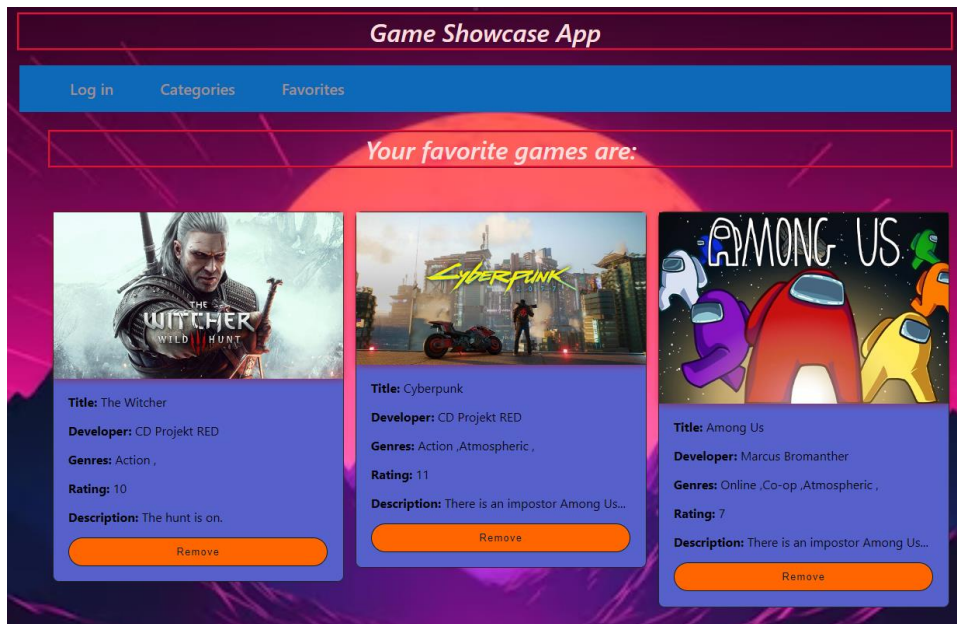
Επιλέγοντας Log in ο χρήστης βρίσκεται στην παραπάνω σελίδα όπου μπορεί να προσθέσει το Username και το Password του ώστε με το Submit να μεταφερθεί στην σελίδα Categories όπου φαίνονται τα παιχνίδια.





Η σελίδα Categories είναι η παραπάνω όπου από τις επιλογές στα αριστερά μπορούν να επιλεγθούν από τον χρήστη κατηγορίες παιχνιδιών και να φιλτραριστούν τα παιχνίδια που προβάλλονται.

Με το κουμπί Favorite ο χρήστης μπορεί να προσθέσει τα παιχνίδια που θέλει στην σελίδα Favorites που ακολουθεί.



Η σελίδα Favorites περιέχει τα παιχνίδια που έχει επιλέξει ο χρήστης και το κουμπί Remove ώστε να αφαιρέσει όποια δεν θέλει πια.

[Μελλοντικές λειτουργίες που θα προστεθούν]

Firestore login: Ο κώδικας για το implementation του login με Google authentication υπάρχει ήδη στο firebase.js αλλά αντιμετώπισα κάποια προβλήματα και για αυτό δεν είναι λειτουργικός ακόμα.

Επίσης σκοπεύω να προσθέσω εναλλακτικά token authentication.

Υπάρχουν κουμπιά που προορίζονται για σύνδεση με social platforms όπως Instagram, LinkedIn και WhatsApp.

Ακόμα, δημιούργησα μία δεύτερη σελίδα Login (LoginPage.js) με διαφορετικά components από React-Bootstrap η οποία θα δουλεύει καλύτερο με το firebase.js.

Τέλος, στο back end υπάρχει βασικός κώδικας για υποστήριξη περισσότερων από έναν χρηστών καθώς και η βάση για Users, ώστε να μπορεί ο κάθε χρήστης να κρατάει την δική του λίστα αγαπημένων.

[Συμπεράσματα]

Η εφαρμογή θα αποτελέσει χρήσιμο εργαλείο οποιουδήποτε ασχολείται με το PC Gaming όπως εγώ.

Μελλοντικές επεκτάσεις της εφαρμογής αποτελούν την προσθήκη νέων παιχνιδιών από τον χρήστη, διαγραφή παιχνιδιών και επεξεργασία απευθείας από το user interface καθώς οι λειτουργίες αυτές είναι ήδη δυνατές μέσω του GameApi αλλά απαιτούν μια σχετική γνώση προγραμματισμού για να μπορέσει ο χρήστης να τις εφαρμόσει με ευκολία.

Θέλω να ευχαριστήσω την καθηγήτρια μου κ.Κωνσταντίνα Χρυσafiάδη για την επίβλεψη της εργασίας.

Σπύρος Καφίρης

Π15055

Πανεπιστήμιο Πειραιώς