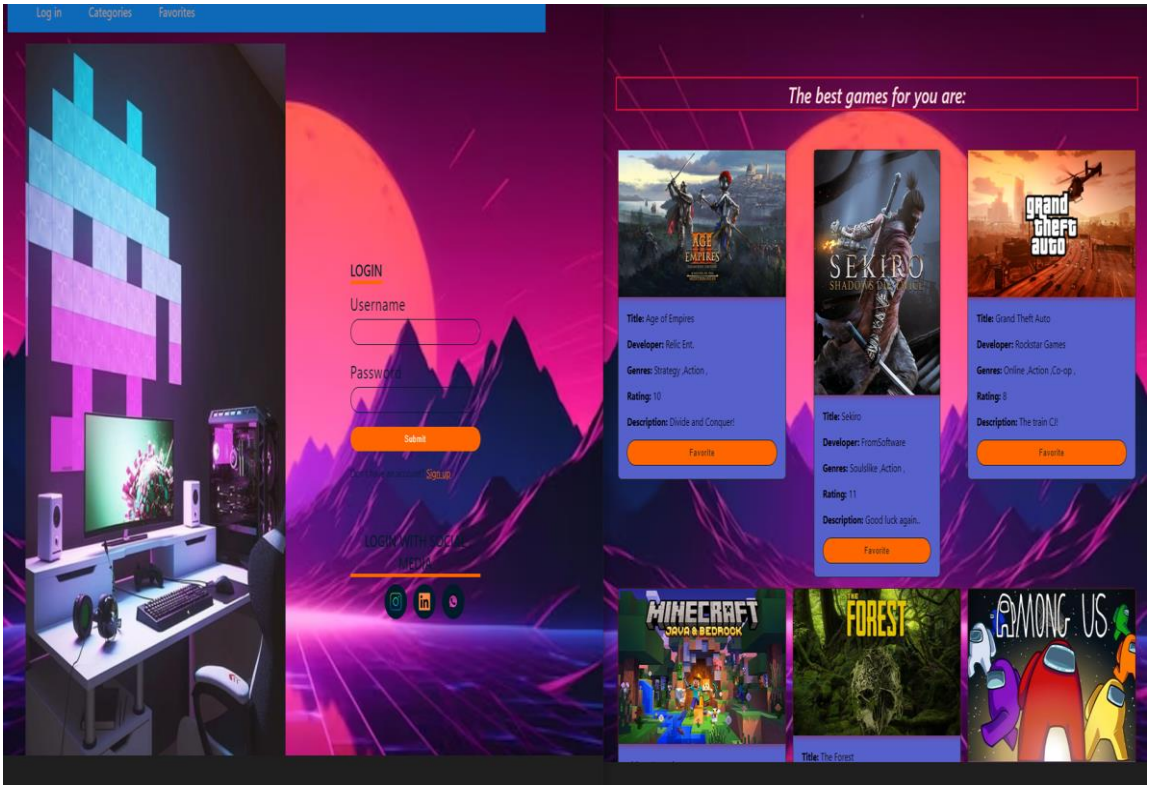


[Game Recommendation App]

[Σπύρος Καφίρης]

[Πτυχιακή]

[3 Νοεμβρίου 2023]

[Περίληψη]

Το αντικείμενο της πτυχιακής εργασίας είναι μια ψυχαγωγική εφαρμογή προτάσεων παιχνιδιών*. Ο χρήστης θα έχει μπροστά του μία λίστα από παιχνίδια με την περιγραφή και τις πληροφορίες τους καθώς και επιλογές για φιλτράρισμα της λίστας και προσθήκη στα αγαπημένα.

[Στόχοι]

- Κάθε άτομο που ψυχαγωγείται στον κόσμο του διαδικτυακού παιχνιδιού πάντα έχει το δίλλημα του τι θα διαλέξει να παίξει σήμερα.

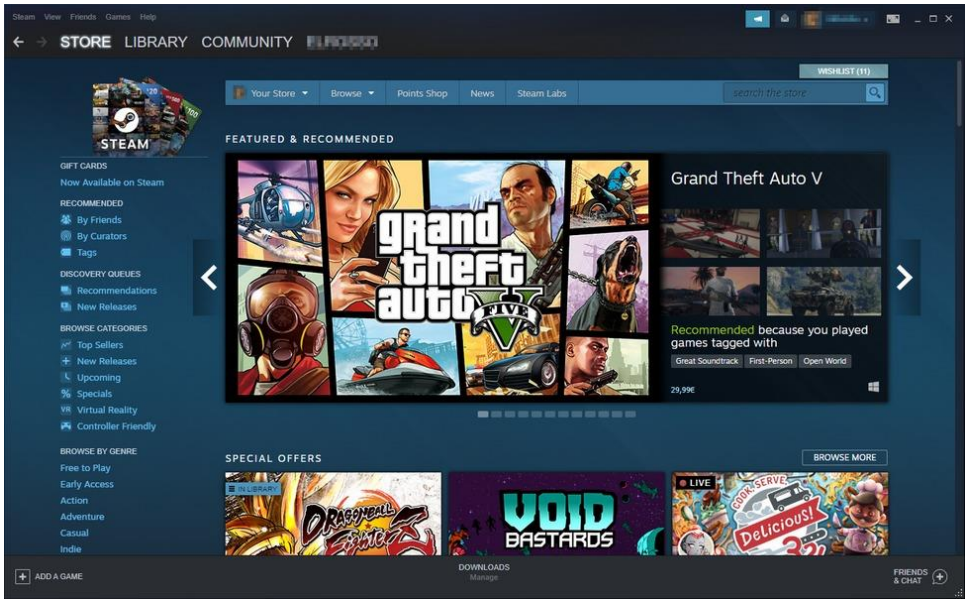
- Αυτή η εφαρμογή βοηθάει σε αυτό ακριβώς το πρόβλημα, προτείνοντας στον εκάστοτε παίκτη τα κατάλληλα παιχνίδια για τις προτιμήσεις του, σε ένα ευχάριστο και γρήγορο περιβάλλον.

[Παρόμοιες Εφαρμογές]

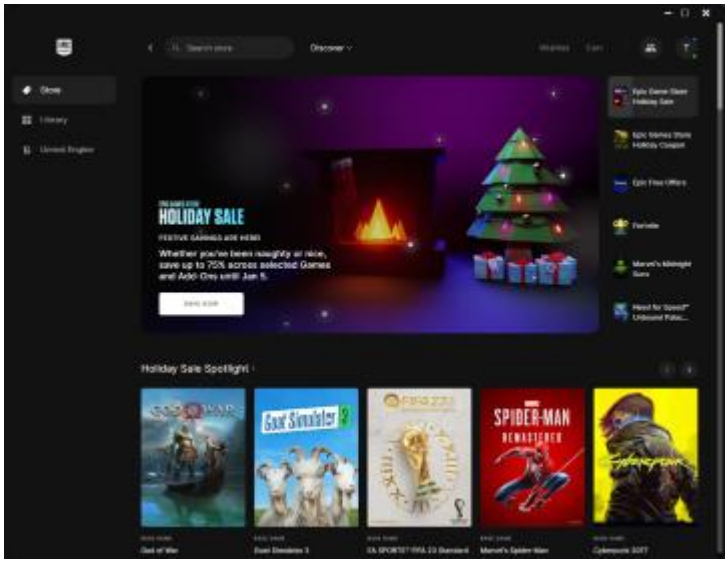
Γνωστές εφαρμογές που βοήθησαν στην ιδέα και υλοποίηση της παρούσας εφαρμογής είναι το Steam, το οποίο σήμερα φιλοξενεί πάνω από 120 εκατομμύριους μηνιαίους χρήστες και πάνω από 50 χιλιάδες παιχνίδια.

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.

Η διαφοροποίηση της εφαρμογής βρίσκεται σε αυτό ακριβώς το κομμάτι καθώς έχει ένα πιο ατμοσφαιρικό user interface με πιο έντονα χρώματα, πιο διαδραστικό καθώς και με περισσότερη πληροφορία για το κάθε παιχνίδι πριν επιλεγεί από τον χρήστη.



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



[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.

Σε αυτήν μπορεί να επιλέξει την/τις κατηγορίες παιχνιδιών που επιθυμεί και η λίστα θα φιλτραριστεί ανάλογα με αυτές ώστε να πάρει προτάσεις μόνο για τις κατηγορίες που θέλει.

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 μπορεί να δει τα επιλεγμένα παιχνίδια που έχει προσθέσει καθώς και να τα αφαιρέσει με το αντίστοιχο κουμπί.

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

[Ανάλυση απαιτήσεων]

Η εφαρμογή είναι σχεδιασμένη για προσωπική χρήση (ένας χρήστης) και απαιτεί 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.

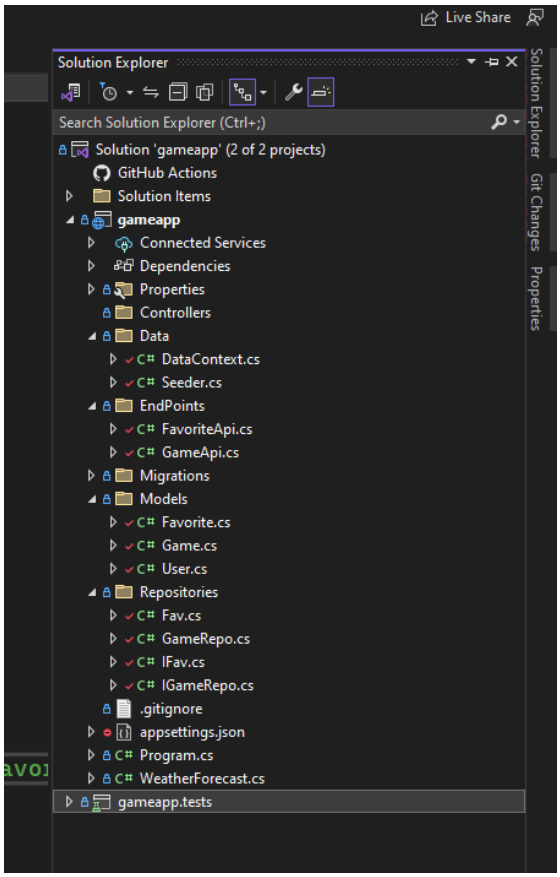
Όλα τα προγράμματα είναι δωρεάν για χρήση.

[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.

{Back-end}

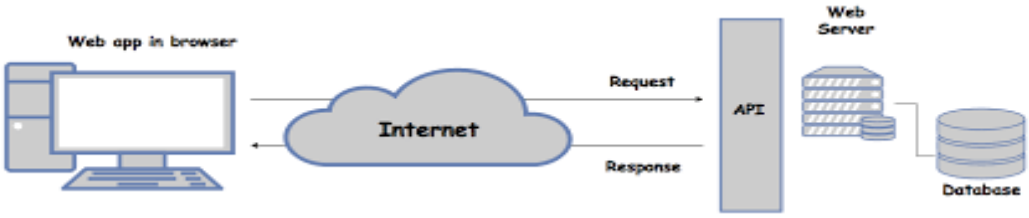
Η δομή του κώδικα σε C# με χρήση Visual Studio είναι η παρακάτω.



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 δεί.

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.

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.

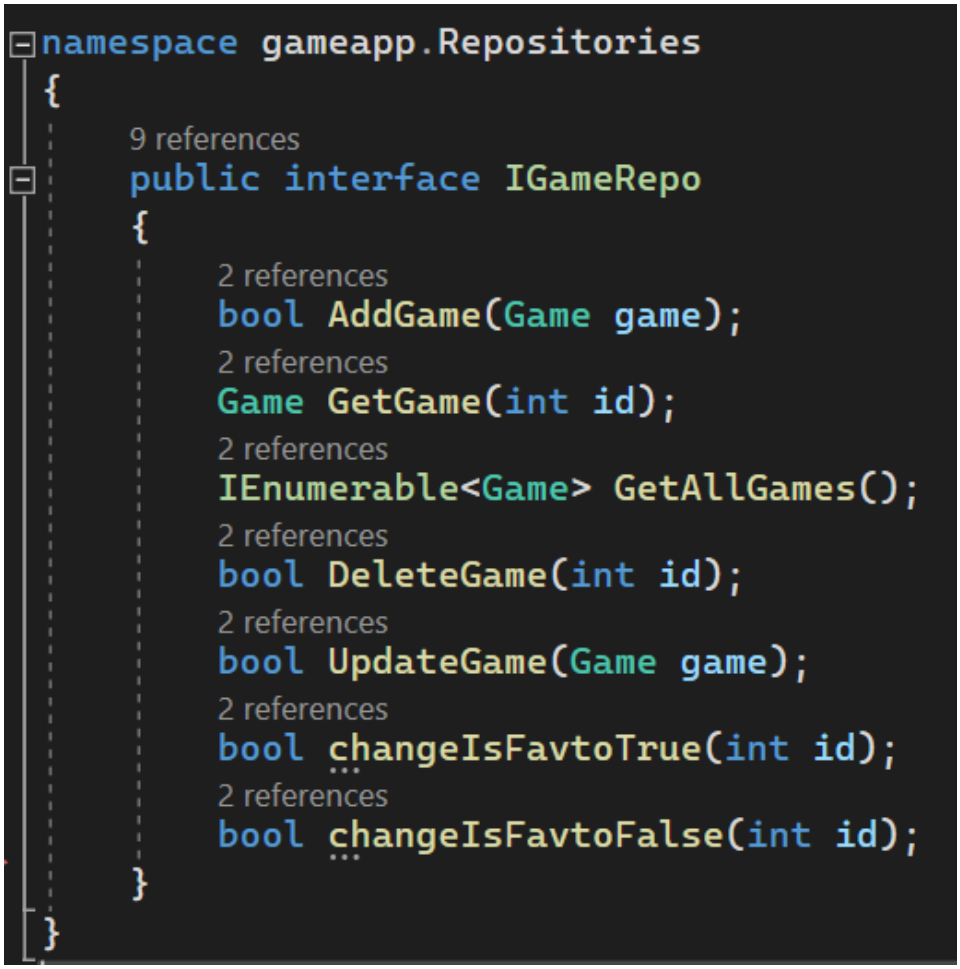
Genres: Η λίστα με τις διάφορες κατηγορίες που ανήκει το κάθε παιχνίδι.

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.



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.

ChangeIsFavtoTrue/changeIsFavtoFalse: αλλαγή της μεταβλητής isFavorite σε true/false αντίστοιχα.

Repositories/GameRepo: Ανάπτυξη των λειτουργιών που προαναφέρθηκαν.

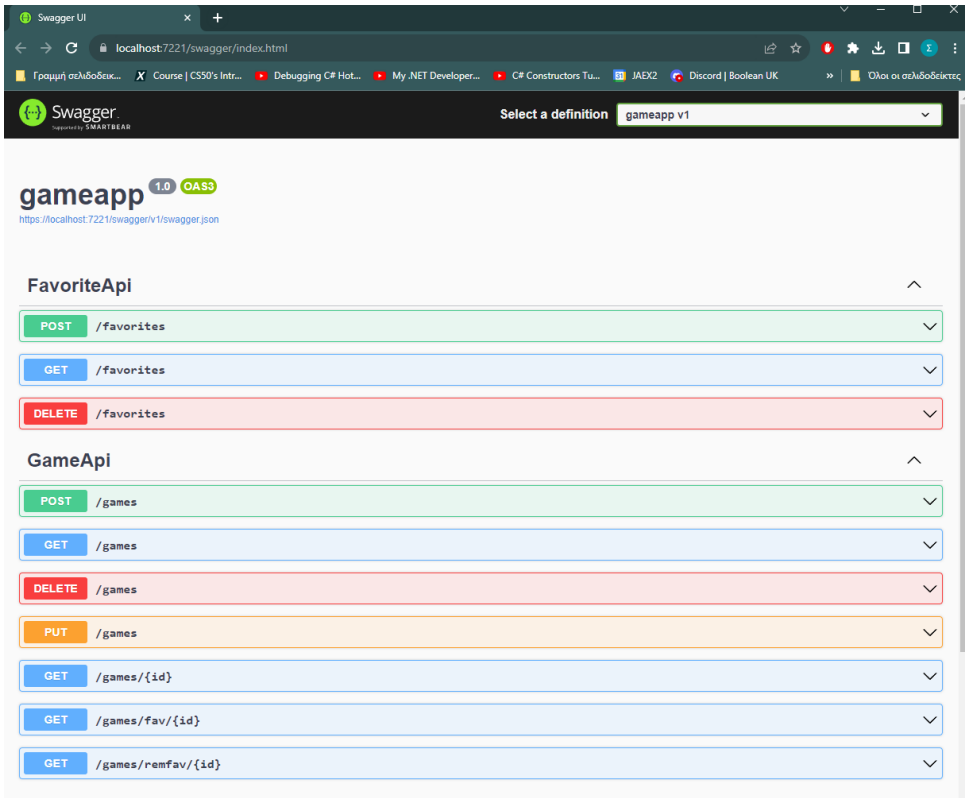
```
2 references
public IEnumerable<Game> GetAllGames()
{
    using (var db = new DataContext())
    {
        return db.Games.ToList();
    }
    return null;
}
```

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.

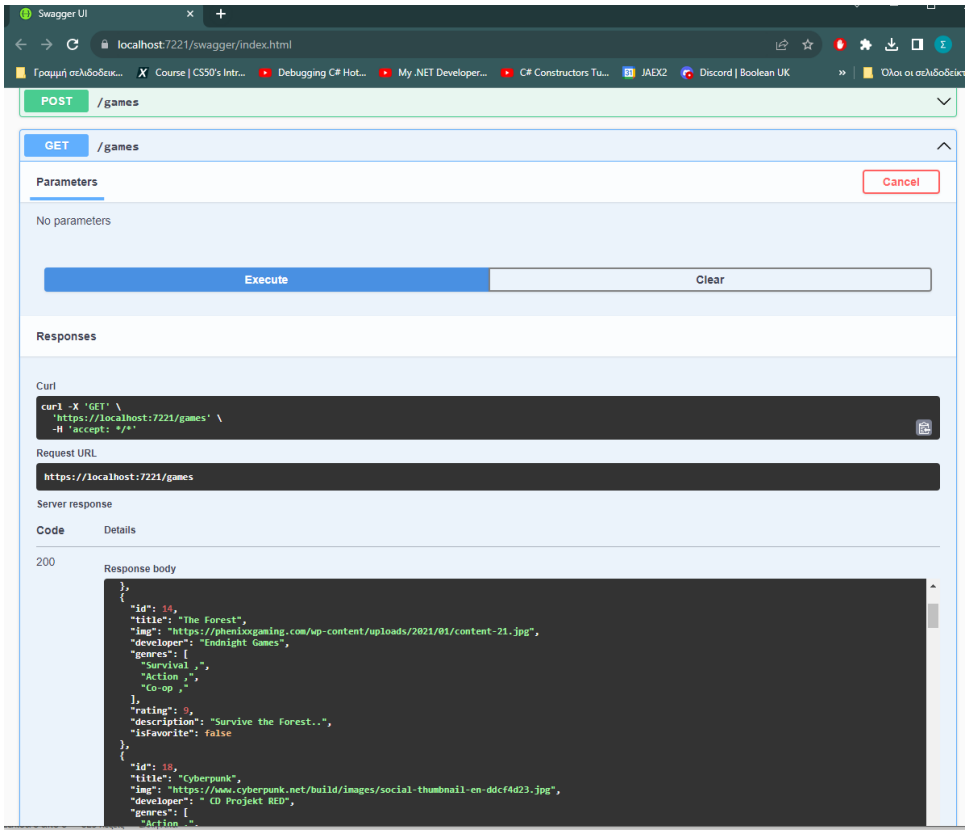
Program.cs: Κλάση αυτόματα δημιουργημένη από την βιβλιοθήκη dotnet όπου προστέθηκαν κάποιες γραμμές κώδικα απαραίτητες για τις παραπάνω λειτουργίες.

Το SwaggerApi που δημιουργήθηκε για την εφαρμογή είναι το παρακάτω.



Το GameApi κάνει όλες τις λειτουργίες που προαναφέραμε.

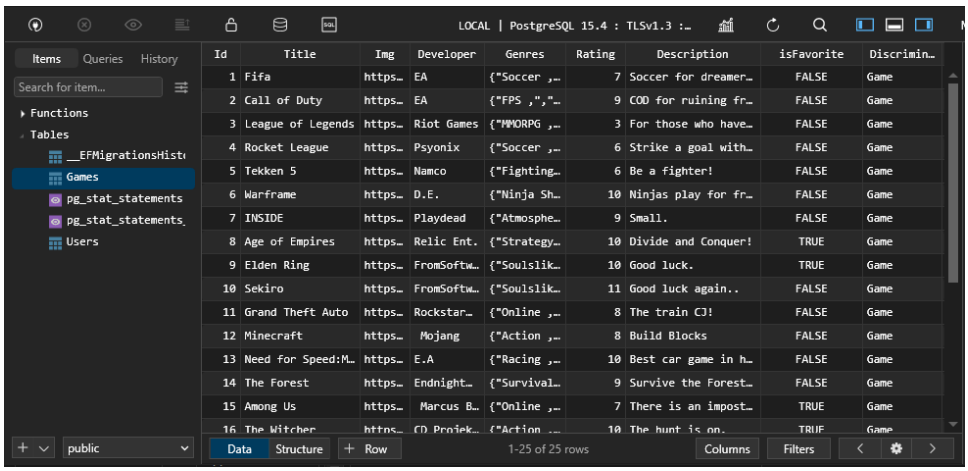
Παραδείγματος χάρη η Get /games που δείχνει στον χρήστη όλα τα παιχνίδια της βάσης.



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

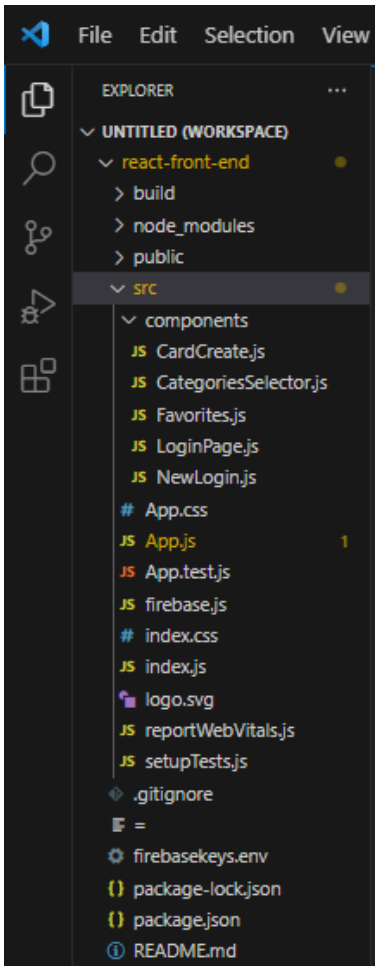
{Βάση Δεδομένων- TablePlus}

Η βάση και ο πίνακας Games με όλα τα δεδομένα.



{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:

```
App.js   NewLogin.js   CardCreate.js   Favorites.js
react-front-end > src > components > NewLogin.js > NewLogin > handleSubmit
12     return(
13     <section>
14         <div className="image">
15             <img src="https://images.squarespace-cdn.com/content/v1/55bebb51e4b036c52ebe8c45/1581577517874-3VCI
               +office+game+room" alt="background image"></img>
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></input>
25                     </div>
26                     <div className="input">
27                         <span>Password</span>
28                         <input type="password" name="password" required></input>
29                     </div>
30                     <div className="input">
31                         <input type="submit" value="Submit" id="submit-button" onClick={handleSubmit}></input>
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"></
                           button>
38                     </div> */}
39                 </form>
```

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 src="https://static.vecteezy.com/system/resources/previews/
                        instagram-icon-logo-free-png.png"></img></li>
44                      <li><img src="https://upload.wikimedia.org/wikipedia/commons/thumb/c
                        640px-LinkedIn_logo_initials.png"></img></li>
45                      <li><img src="https://upload.wikimedia.org/wikipedia/commons/thumb/5
                        png"></img></li>
46                  </ul>
47              </div>
48          </div>
49      </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 όποτε 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}/>
```

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

```
function CardCreate() {
    const [gameData, setGameData] = useState([]);
    const [filterTags, setFilterTags] = useState([])

    useEffect(() => {
      axios.get("https://localhost:7221/games").then((response) => {
        //console.log(response.data)
        setGameData((existingData) => {
          return response.data;
        });
      });
    }, []);

    const filteredDATA = gameData.filter((game) =>
    filterTags.length > 0
      ? filterTags.every((filterTag) =>
          game.genres.includes(filterTag)
        )
      : gameData
    )

    const handleChange = (event) => {
      if (event.target.checked) {
        setFilterTags([...filterTags, event.target.value])
      } else {
        setFilterTags(
          filterTags.filter((filterTag) => filterTag !== event.target.value)
        )
      }
    }
```

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 ‘καρτες’.

Σειρές 29-37: προσθήκη κατηγορίας παιχνιδιων 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 μαζι 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.

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

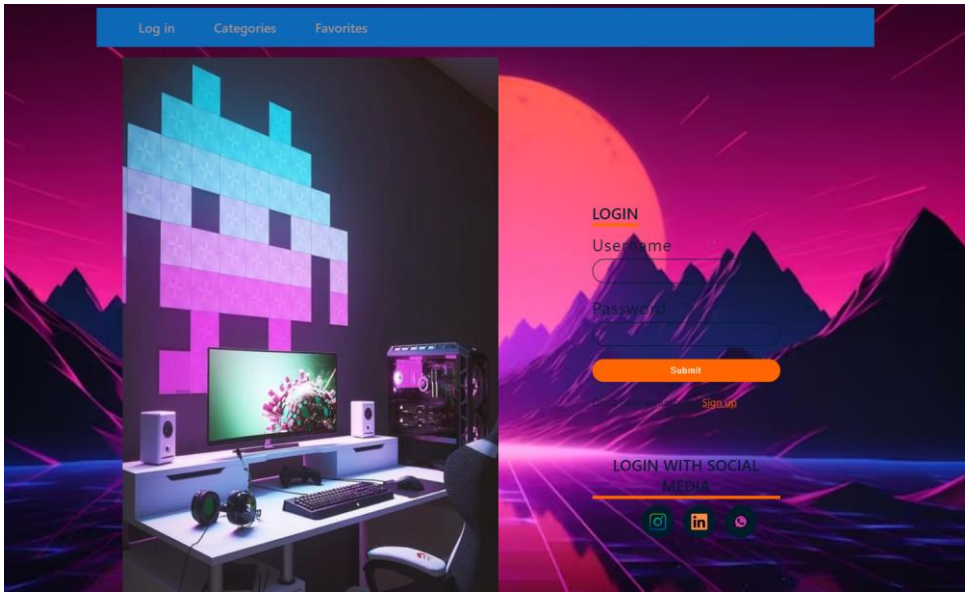
Σειρές 20- : ομοίως με addtofav του CardCreate για αλλαγή του isFavorite σε false και αφαίρεση του παιχνιδιού από τα Favorites.

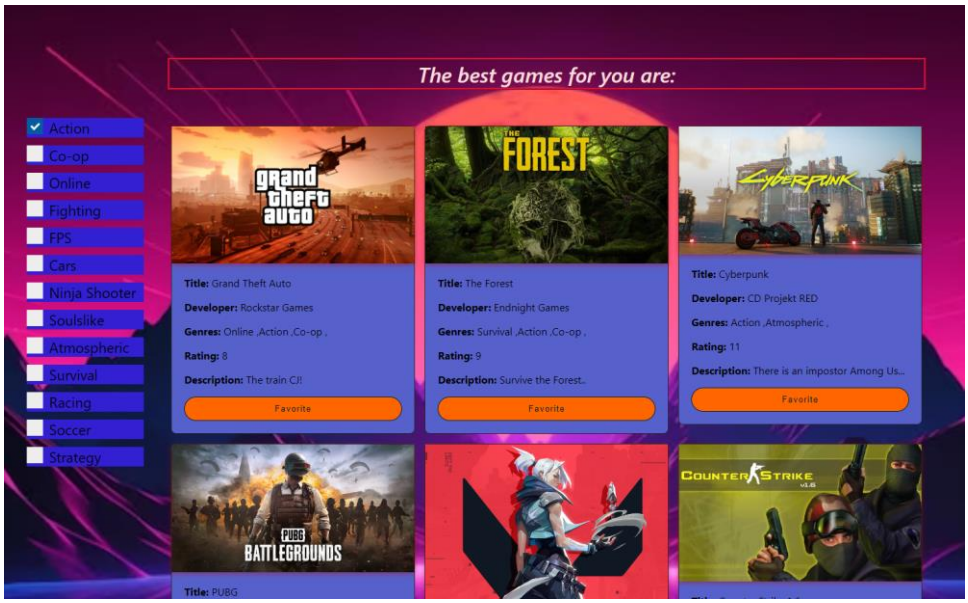
```
<h1>Your favorite games are:</h1>
<Row xs={1} md={3} className="cards">
  {
    gameData.filter((game) => game.isFavorite==true).map(filteredGames => (
    <Col key={filteredGames.id}>
      <Card>
        <Card.Img
        className="card--img" variant="top" src={filteredGames.img} />
        <Card.Body 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>
```

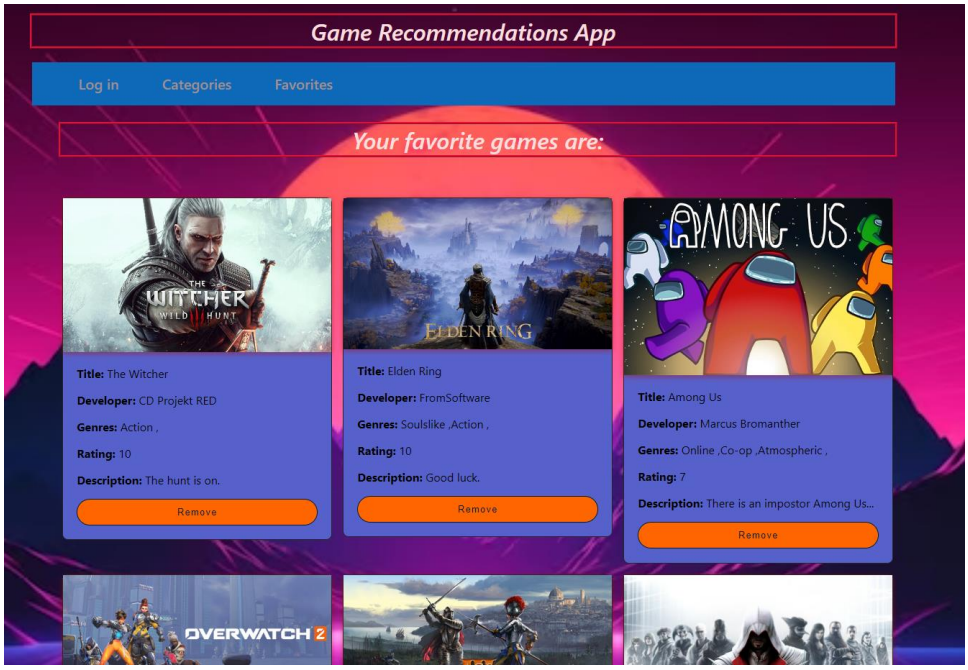
Δημιουργία ‘καρτών’ για προβολή των Αγαπημένων παιχνιδιών του χρήστη και κουμπί Remove για αφαίρεση αυτών.

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

[Game Recommendation App τελική εφαρμογή]







[Τεχνολογίες που χρησιμοποιήθηκαν]

Για το back-end χρησιμοιποιήθηκε C#.

Για την βάση δεδομένων ElephantSql και TablePlus.

Για το front-end χρησιμοποιήθηκαν Html, Javascript, CSS, Bootstrap και React.

[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.

Επίσης, υπάρχουν διαφορετικές μορφές της σελίδας Login (LoginPage) με άλλες λειτουργίες και επίσης ο κώδικας για την σύνδεση με Gmail (firebase).

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

Σπύρος Καφίρης

Π15055

3 Νοεμβρίου 2023

Πανεπιστήμιο Πειραιώς