

**SAPIENTIA ERDÉLYI MAGYAR TUDOMÁNYEGYETEM  
MAROSVÁSÁRHELYI KAR,  
INFORMATIKA SZAK**



**SAPIENTIA  
ERDÉLYI MAGYAR  
TUDOMÁNYEGYETEM**

Cronotus - Sportesemény koordináló webalkalmazás

**DIPLOMADOLGOZAT**

Témavezető:

Dr. Jánosi-Rancz Katalin Tünde, Ábrám Szilveszter Levente  
Egyetemi adjunktus

Végzős hallgató:

**2024**

**UNIVERSITATEA SAPIENTIA DIN CLUJ-NAPOCA  
FACULTATEA DE ȘTIINȚE TEHNICE ȘI UMANISTE,  
SPECIALIZAREA INFORMATICĂ**



**UNIVERSITATEA  
SAPIENTIA**

Cronotus - Aplicație web pentru coordonarea evenimentelor sportive

**LUCRARE DE DIPLOMĂ**

Coordonator științific:

Dr. Jánosi-Rancz Katalin Tünde, Ábrám Szilveszter Levente  
Lector universitar

**2024**

Absolvent:

**SAPIENTIA HUNGARIAN UNIVERSITY OF  
TRANSYLVANIA**  
**FACULTY OF TECHNICAL AND HUMAN SCIENCES**  
**COMPUTER SCIENCE SPECIALIZATION**



**SAPIENTIA**  
HUNGARIAN UNIVERSITY  
OF TRANSYLVANIA

Cronotus - Web application for organizing sporting events

**BACHELOR THESIS**

Scientific advisor:

Dr. Jánosi-Rancz Katalin Tünde, Ábrám Szilveszter Levente  
Lecturer

Student:

**2024**

### **Declarație**

Subsemnatul/a Abram Silvester Revete absolvent(ă) al/a specializării Informatică, promoția 2024 cunoscând prevederile Legii învățământului superior nr. 199 din 2023 și a Codului de etică și deontologie profesională a Universității Sapientia cu privire la furt intelectual declar pe propria răspundere că prezenta lucrare de licență/proiect de diplomă/disertație se bazează pe activitatea personală, cercetarea/proiectarea este efectuată de mine, informațiile și datele preluate din literatura de specialitate sunt citate în mod corespunzător.

Localitatea, Târgu Mureș  
Data: 20.06.2024

Absolvent  
Semnătura.....Abram Silvester Revete

UNIVERSITATEA „SAPIENTIA” din CLUJ-NAPOCA  
Facultatea de Științe Tehnice și Umaniste din Târgu Mureș  
Specializarea: **Informatică**

**Viza facultății:**

**LUCRARE DE DIPLOMĂ**

Coordonator științific: <b>Lector dr. Jánosi-Rancz Katalin-Tünde</b>	Candidat: <b>Abram Szilveszter Levente</b> Anul absolvirii: <b>2024</b>
---	--

**a) Tema lucrării de licență:**

Cronotus - Aplicație web pentru coordonarea evenimentelor sportive

**b) Problemele principale tratate:**

- Listarea evenimentelor sportive
- Organizarea evenimentelor sportive

**c) Desene obligatorii:**

- Diagrame privind structura sistemului realizat
- Diagrame UML

**d) Softuri obligatorii:**

- React library, Vite
- ASP .NET Core, Entity Framework Core, Microsoft SQL Server

**e) Bibliografia recomandată:**

- Ana Brochado, Pedro Dionisio, Organizing Sporting Events: the promoters' perspective, Journal of Business and Industrial Marketing, 2021
- S. Warner, E. Sparvero, S. Shapiro, A. Anderson, Yielding healthy community with sport?, Journal of Sport for Development, 2017

**f) Termene obligatorii de consultații: săptămânal**

**g) Locul și durata practicii:** Universitatea „Sapientia” din Cluj-Napoca,  
Facultatea de Științe Tehnice și Umaniste din Târgu Mureș

Primit tema la data de: 1 mai 2023

Termen de predare: 24 iunie 2024

Semnătura Director Departament

Semnătura responsabilului  
programului de studiu

Semnătura coordonatorului

Semnătura candidatului

# Kivonat

A mozgás és a különböző sportok üzése szerves részét képezik az emberek minden-napi életének. Ahhoz, hogy egészségesen élhessünk, mindenkor más és más sportokat választunk, melyekbe naponta energiát és időt fektetünk. A mozgás nem csak fizikai, de szellemi egészségünkre is nagy hatással van, továbbá az, hogy kikkel sportolunk, nagymértékben meghatározza a mozgással eltöltött időnk minőségét[25]. mindenki számára fontos a közösség, úgy a magánéletben, mint a sportolásban is. A legnagyobb kihívást sokszor az jelenti, hogy megtaláljuk az ideális társaságot és a közösen eltöltött idő valóban minőségi legyen.

Dolgozatom célja, egy olyan alkalmazás bemutatása, amely a sportolásra ösztönöz mindenkit és segít a sportolói közösségg kialakításában is.

A Cronotus egy olyan internetes felületet biztosít, ahol az emberek könnyedén kiküszöbölnétek a sportoláshoz szükséges társaság hiányában felmerülő problémát. Lehetőséget biztosít sportesemények szervezésére, illetve ezek bongészésére, hogy biztosak lehessenünk abban, hogy többé senki nem marad otthon azért, mert nem volt akivel sportolni menjen. A projekt továbbá szem előtt tartja azt is, hogy egyes események könnyedén követhetőek legyenek, lehetőséget biztosít a szervezőknek arra, hogy naprakész információkkal lássák el a sportolókat, ami segíthet a közösségek abban, hogy a számukra legmegfelelőbb eseményeket találják meg.

Kulcsszavak: sport, sportesemény, szervezés, koordinálás, közösség

# Rezumat

Exercițiile fizice și practicarea diferitelor sporturi fac parte integrantă din viața de zi cu zi a oamenilor. Pentru a avea o viață sănătoasă, cu toții alegem diferite sporturi, în care investim energie și timp în fiecare zi. Exercițiile fizice nu au numai beneficii fizice, ci au efecte semnificative și asupra sănătății mintale, iar oamenii cu care facem sport pot determina calitatea timpului pe care îl petrecem făcând mișcare.<sup>[25]</sup> Comunitatea este importantă pentru toată lumea, atât în viața personală, cât și în activitățile sportive. Cea mai mare problemă este adesea aceea de a găsi compania potrivită și de a face timpul petrecut împreună cu aceasta să fie cu adevărat de calitate.

Scopul tezei este de a prezenta o aplicație care încurajează oamenii să facă sport tuturor și, de asemenea, ajută la construirea unei comunități de sportivi.

Cronotus oferă o platformă online prin care oamenii pot cu ușurință să depășească problema de a nu avea companie pentru a face sport. Aplicația vă permite organizarea, căutarea evenimentelor sportive, astfel încât să ne asigurăm că nimeni nu rămâne acasă pentru că nu are cu cine să iasă la sport. Proiectul își propune, de asemenea, să facă anumite evenimente ușor de urmărit, oferindu-le organizatorilor posibilitatea de a oferi informații actualizate sportivilor, care să ajute comunitatea să să găsească evenimentele care sunt cele mai potrivite pentru ei.

Cuvinte cheie: sport, eveniment sportiv, organizare, coordonare, comunitate

# Abstract

Physical activity and practicing various sports are an integral part of people's everyday lives. In order to live a healthy life, we all choose different sports, in which we invest energy and time every day. Physical exercise has proven to not only have physical benefits, but also significant effects on mental health, furthermore the people we do sports with can determine the quality of the time we spend exercising.[25] Choosing the right community for doing sports and everyday activities is a challenge that everyone faces, since it great impact on our lives.

The purpose of this thesis is providing a platform that serves as a middleware for people that want to connect with others, and fight the problem of not having company to exercise.

Cronotus provides an online platform where people can easily overcome the problem of not having company to exercise. The application allows one to organize, search for sports events, so that we can make sure that no one is left home because they had no one to go out with. The project also aims to make certain events easy to follow, providing organizers with the opportunity to provide up-to-date information to athletes, which can help the community to find the events that are most suitable for them.

Keywords: sport, sporting event, organization, coordination, community, social

# Tartalomjegyzék

<b>1. Bevezető</b>	<b>10</b>
<b>2. A Cronotus projekt</b>	<b>11</b>
2.1. Piackutatás . . . . .	11
2.2. A Cronotus funkcionalitásai . . . . .	13
2.3. Szerepkörök . . . . .	13
2.3.1. A szervező . . . . .	13
2.3.2. A játékos . . . . .	14
<b>3. Felhasznált technológiák</b>	<b>16</b>
3.1. A szerver . . . . .	16
3.1.1. A rétegelt architektúra előnyei . . . . .	17
3.1.2. Az adatbázis . . . . .	19
3.1.3. A Cronotus API . . . . .	19
3.1.4. Hitelesítés és Engedélyezés . . . . .	20
3.1.5. Globális hibakezelés . . . . .	22
3.2. A felhasználói felület . . . . .	22
3.2.1. Felhasznált technológia . . . . .	22
3.2.2. Kommunikáció a szerverrel . . . . .	23
3.2.3. Oldalak közötti navigálás . . . . .	24
3.2.4. Munkamenetkezelés JWT tokenekkel . . . . .	26
<b>4. A Cronotus felhasználói felülete</b>	<b>28</b>
4.1. A fogadó oldal és a regisztráció . . . . .	28
4.2. A profil oldal . . . . .	30
4.3. A sportesemény böngésző . . . . .	31
4.3.1. Sportesemények szűrése . . . . .	35
4.4. Lapszámozás - Pagination . . . . .	37
<b>5. A projekt menedzselése</b>	<b>38</b>
5.1. Verziókezelés Git segítségével . . . . .	38

# 1. fejezet

## Bevezető

Annak ellenére, hogy az információ közlése és elérése napjainkban rendkívül gyors és olcsó, mégis előfordul, hogy alapvető tevékenységeket igen csak maradi módon kezelnek az emberek. Többek között idetartozik egyes sportesemények koordinálása is. A résztvevők gyakran hajlamosak egy nem erre a célra készített platformra hagyatkozni, ami természetesen nem minden működik úgy, ahogy ideális lenne. Ennek eredményeként megötörtenik, hogy lemaradnak az emberek olyan információkról, ami egyébként releváns lett volna számukra.

A Cronotus projekt ennek a problémának a megoldását egy ingyenes és könnyen kezelhető platform biztosításával közelíti meg, ami arra törekszik, hogy kiküszöbölje az esetleges információk elvesztését, vagy akár figyelmen kívül hagyását.

A dolgozat elsősorban bemutatja a projekt általános működését, párhuzamba helyezi más ehhez hasonló internetes platformokkal, szemünk elé tárja az esetleges előnyöket és hátrányokat más szolgáltatásokkal szemben.

A továbbiakban a dolgozat ismerteti velünk mélyebben a Cronotus szerkezetét, ahol felépítés szerinti rétegekre bontja, majd ezeket részletekbe bocsátva tárgyalja. Bemutatja azt, hogy hogyan működik a web kliens-t kiszolgáló szerver, milyen biztonsági intézkedéseket biztosít, illetve hogyan kezeli a felmerülő hibákat. Betekintést biztosít abba, hogy hogyan érkezik el a kívánt információ az adatbázistól egészen a felhasználó felületig.

A következő fejezetek segítségével szeretném bemutatni a felhasznált eszközöket és módszereket, melyek elősegítik a Cronotus helyes működését. Végezetül egy következetést vonok le, illetőleg továbbfejlesztési lehetőségeket tárok az olvasó elé, melyek fényt derítenek a projekt jövőbeli élettartamára.

A platform készítése 2023 októberében kezdődött a Codespring szoftverfejlesztő cég koordinálásával. Megemlíttendő, hogy a felhasznált technológiák visszatükrözik azt a tudást, amit a Marosvásárhelyi Codespring Mentorprogram keretein belül sajátíthattam el, ami 2022 őszi időszakában vette kezdetét. A projektet a koordináló cég segítségével egyedül fejlesztettem.

Az alkalmazás neve, azaz a Cronotus, egy összetett lingvisztikai ihletből merített szó. A név első fele a görög mitológiai Krónosztól származik, akit az idő isteneként ismerünk, míg a név második fele a „mozgás” szó latin fordításából, a ’motus’ szóból ered.

## **2. fejezet**

# **A Cronotus projekt**

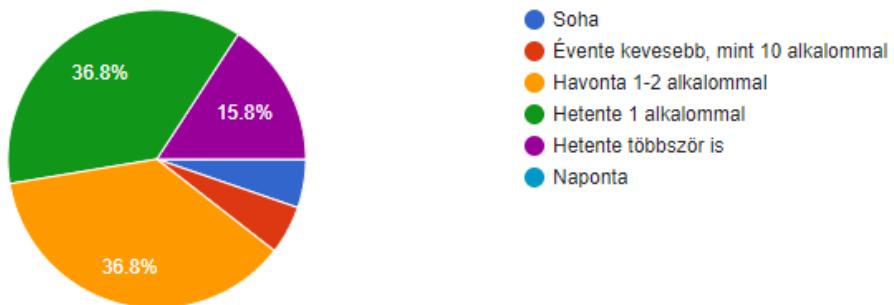
A Cronotus projekt célja egy olyan internetes felület biztosítása, ahol az emberek könnyedén megtalálhatják a számukra legmegfelelőbb sportolási lehetőségeket. Mindez egy webalkalmazás formájában valósul meg, ahol a felhasználók ként regisztrálva tudunk interakcióba lépni az elérhető sporteseményekkel, szervezőkkel.

A rendszer igyekszik azt elérni, hogy különböző módokon hozza közelebb egymáshoz a sportolni vágyó felhasználókat. Abban az esetben, ha létezik egy megfelelő sportesemény könnyedén csatlakozni lehet hozzá, ha meg nem, akkor lehetőség van egy megfelelő új esemény létrehozására.

### **2.1. Piackutatás**

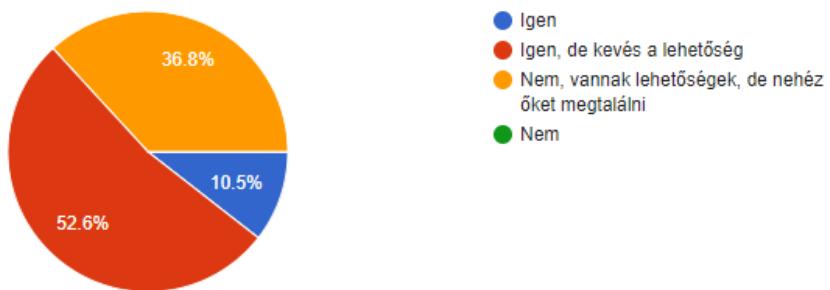
Ahhoz, hogy jobban megérthessük a társaság hiánya okozta sportolási nehézségeket, az alábbiakban egy általunk végzett felmérést fogunk megvizsgálni. A felmérésben összegyűjtött adatok legfőképpen a erdélyi magyar fiataloktól származnak, korosztály szempontjából legjellemzőbben a 19-25 éves korig terjedő egyénekre vonatkoznak. A kutatáson 30 fő vett részt.

Első sorban megbizonyosodtunk arról, hogy a fiatalok igenis érdeklődnek a sportok iránt. A felmérés kimutatta azt, hogy az egyének 90%-a havonta legalább két alkalommal sportol, 36%-uk heti szinten, míg 15%-uk hetente többször is, ahogy ez a 2.1-es ábrán is látható.



**2.1. ábra.** Mennyit sportolnak az emberek?

A továbbiakban a felmérés azt is kimutatta, hogy sok esetben nincs elég sportolási lehetőség az egyének számára. A választ adók 52%-a azt mondta ki, hogy bár könnyen megtalálja a sporteseményeket, nincs elég lehetőség, amelyek közül válogatni lehet, míg az emberek 36%-a azt mondja, hogy nehéz megtalálnia az elérhető sportolási lehetőségeket, ahogy ez látható a 2.2-es ábrán is.



**2.2. ábra.** Könnyen talál sportolási lehetőségeket?

A választ adók 100%-ban egyetértettek abban, hogy hasznukra lenne egy olyan webalkalmazás, amely közelebb vinné őket a sportolási lehetőségekhez, és elérhetővé tenné ezeket számukra, továbbá abban is teljes egyetértés volt, hogy nem ismernek más olyan alkalmazásokat, amely hasonló szolgáltatást nyújtana számukra.

A felmérés arra is rámutatott, hogy az emberek azon része, akik szoktak sporteseményeket szervezni, az esetek nagyrészében olyan felületet használnak, amely nem kifejezetten erre a célra készült. Legnépszerűbb megoldások közé tartozik a Facebook Messenger, WhatsApp, illetve más chat alapú alkalmazások használata. Ez egy egyszerű és gyors megoldás lehet, viszont könnyen elveszhetnek az információk, és a szervezőknek nehéz lehet a résztvevőkkel való kommunikáció.

A felmérésből levonható az, hogy a Cronotus projekt egy igényelt webalkalmazás, amely sok ember számára megoldást jelenthet a sportolással kapcsolatos nehézségekre nézve, így hozzájárulva a közösség szellemi és fizikai egészségéhez egyaránt.

## 2.2. A Cronotus funkcionalitásai

A Cronotus webalkalmazása a következőkkel szolgálhat a felhasználók számára:

- Regisztráció
- Bejelentkezés
- Felhasználói adatok listázása
- Felhasználói adatok szerkesztése
- Felhasználói profilkép és borítókép feltöltése, törlése
- Létrehozott sportesemények listázása
- Létrehozott sporteseményekre való jelentkezés, leiratkozás
- Saját sportesemény létrehozása előre definiált sportok alapján
- Saját sporteseményekhez való képek feltöltése, törlése
- Sportesemények törlése

## 2.3. Szerepkörök

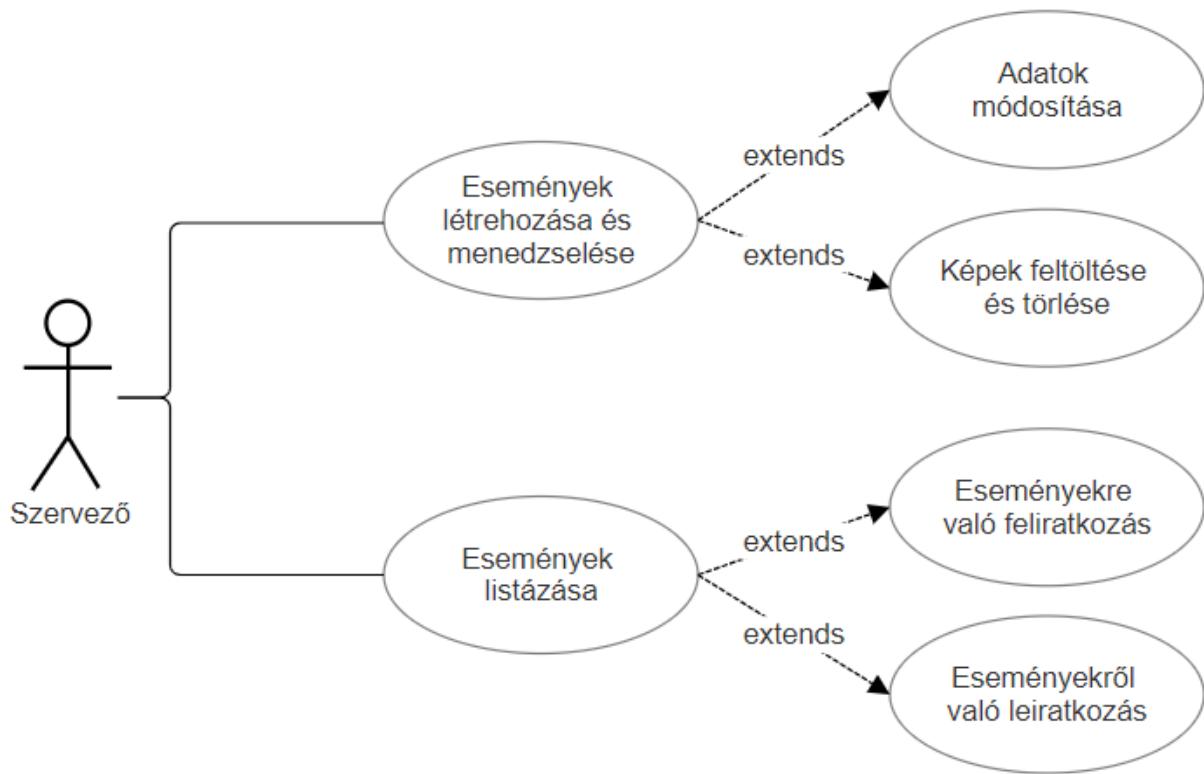
A Cronotus két fő szerepkör elkülönítésével közelíti meg a tárgyalt problémát. Az egyik az „Organizer”, azaz szervezői, a másik pedig a „Player”, avagy játékos szerepkör formájában mutatkozik meg. Egy új felhasználó regisztrálásakor automatikusan mindenki szerepkör jóváíródik a felhasználóhoz, így a webalkalmazás lehetőséget nyújt arra, hogy első perctől könnyedén végezhető legyen minden kívánt tevékenység.

### 2.3.1. A szervező

Ahogyan az a 2.3-as ábrán is látható, a szervező szerepkörnek lehetősége van új sportesemények létrehozására, az ide tartozó információk könnyed kezelésére, valamint a résztvevőkkel való kommunikációra. Továbbá mindenkihez, hogy eseményeket hozhat létre, lehetősége van meglévő eseményekre jelentkezni is, mint bármely más felhasználónak.

### **2.3.2. A játékos**

Amennyiben egy felhasználó csak arra szeretné használni a felületet, hogy sportolási lehetőségeket találjon, úgy a „Player” szerepkör lehetőségei tökéletesen elegendőek számára. A játékos felhasználók, a szervezőkhöz hasonlóan listázhatsák a már létező eseményeket, jelentkezhetnek ezekre, illetve figyelemmel kísérhetik az eseményekkel kapcsolatos információkat.

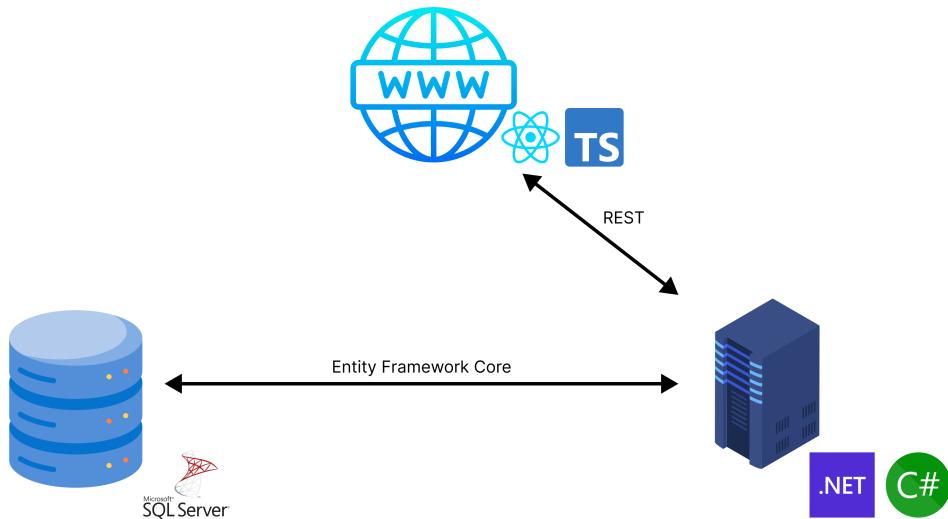


**2.3. ábra.** A szervező szerepkör

## 3. fejezet

# Felhasznált technológiák

A Cronotus alkalmazás két jól elkülöníthető részre bontható fel. Amint a 3.1-es ábrán is látható, a felhasználói felület weben keresztül érhető el, mely a React[15] technológia segítségével lett megalkotva TypeScript[22] nyelven, míg az alkalmazás szerver oldali része egy .NET keretrendszeren[12] alapuló program, mely C# nyelven[2] lett megírva és az Entity Framework Core[6] által lefektetett szabályrendszert használva kommunikál egy Microsoft SQL Server[20] adatbázissal, ahol minden adat eltárolásra kerül. Ahhoz, hogy ezeket az adatokat a felhasználó elérhesse, a kliens és szerver oldal között egy REST API[18] segítségével történik meg a kommunikáció.

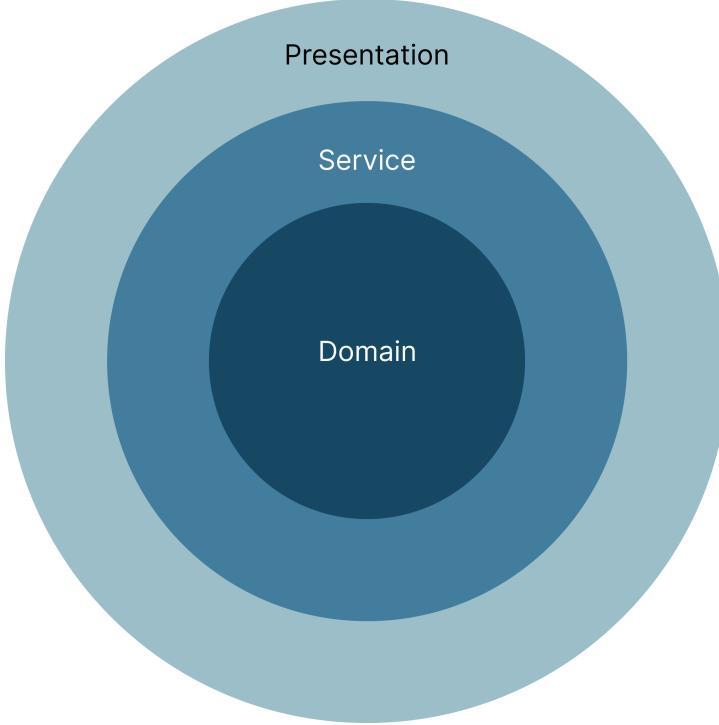


**3.1. ábra.** Rendszer architektúra komponensekre bontva

### 3.1. A szerver

A szerver feladata az, hogy kiszolgálja a webalkalmazásról beérkező kéréseket. Ez a feladat alapszinten viszonylag könnyen elvégezhető lenne, viszont nem elég csak csupán elvégezni, hanem a legmegfelelőbb módon kell ezt véghez vinni. Fontos szem előtt tartani

a biztonságot, kérésenkénti időt, illetve megbízhatóságot. Az első fontos döntés a szerverrel kapcsolatban az, hogy milyen architekturát valósít meg. A Cronotus esetében egy rétegelt architektúrai felépítésről[13] beszélhetünk, mely három fő elemre bontja a szerver működését.



**3.2. ábra.** A rétegelt szerver architektúrája

A három réteg a 3.2-es ábrán feltüntetett módon helyezkedik el. A legalsó szint a Domain, melynek mindenekkel annyi a feladata, hogy az adatbázissal kommunikáljon. Ezen a szinten történik meg az adatok kinyerése, vagy akár az alapszintű műveletek véghezvitelére, például létrehozás, vagy törlés.

A kör középpontjától kifelé haladva a következő a Service réteg. Ez az a szint, amely a biznisz logikát végre hajta a kapott adatokon. Abban az esetben, ha a kliens oldal egy kérést küld valamely adat lekérésére, úgy a kéréshez szükséges eredmény ezen a szinten formálódik olyan alakba, amely megfelelő a fogyasztónak. A szint felelősége úgy is felfogható, mint egy tolmacs, mely a kliens és Domain között működik. Főbb feladatai a komplex struktúrájú adatok egyszerű alakra és utasításokra bontása, illetve egyszerű adatok előkészítése és prezentálható formába szervezése a kliens számára.

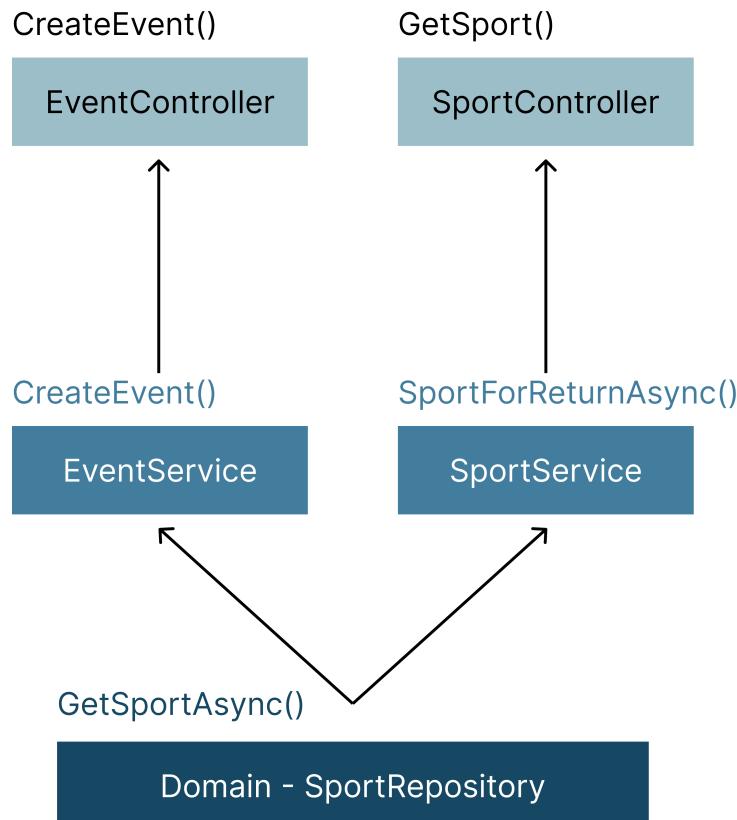
A legfelső, a Presentation réteg feladata a fogyasztó irányába történő kérésekkel kapcsolatos visszajelzések megfelelő kezelése, illetve a Service rétegtől kapott adatok átirányítása a helyes HTTP kóddal.

### 3.1.1. A rétegelt architektúra előnyei

Felmerülhet a kérdés, hogy miért érdemes az előbbiekbén [3.2] bemutatott formába szervezni a szerver felépítését. Ahhoz, hogy ezt bemutassuk, fontos először megvizsgálni

az architektúrán belüli egyes rétegek közti függőségek folyását. Ha ezt a felépítést követjük, akkor a függőségek a legbelő szintektől a felsőbb szintek irányába folynak, azaz a Domain réteg nem függ semmítől. Ez betudható szabályként is, tehát egy réteg minél alacsonyabban van a felépítésben, annál kevesebb függősége van. Az előbbiekbén leírt függőségi irányzat úgy is nevezhető, mint az úgynevezett Inversion of Control (IoC) [8].

Továbbá minden réteg csak kizárolag az alatta levő rétegre hivatkozhat. Ez azért hasznos, mert így nem kell aggódunk afelől, hogy hogyan fog egy felsőbb szintű művelet implementálni egy alacsonyabb osztályút, elég kontraktokat felállítanunk, melyeket teljesíthetnek a felső rétegek. Ez fontos tulajdonsága az architektúrának, hiszen nagy szabadságot nyújt egyes szintek elkészítésében, hiszen elég csak az oda tartozó logikára figyelnünk.



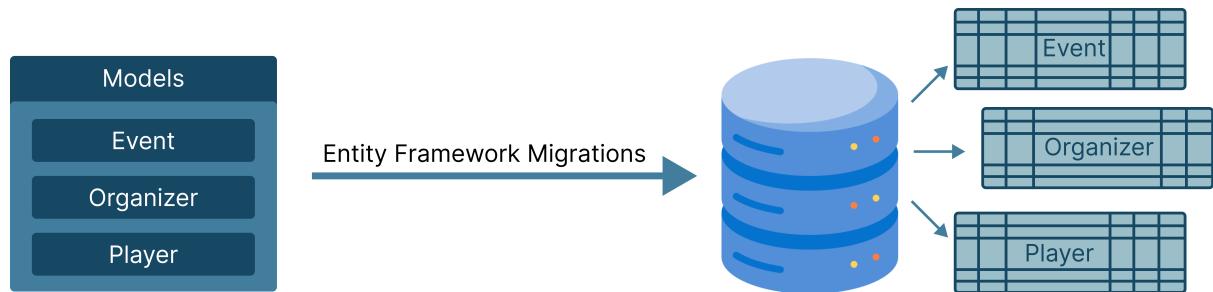
**3.3. ábra.** A SportRepository egy Domain szintű műveletének eredményének kétféleképpen való felhasználása felsőbb szinteken

A 3.3-as ábrán látható egy példa, mely a rétegelt architektúra egy előnyét használja fel. A SportRepository-ban rendelkezésünkre áll egy függvény, mely egy Sport Entity-t térít vissza. A Repository-nak ebben az esetben nem kell tudnia, hogy hogyan fogják a felsőbbrendű rétegek felhasználni az innen kapott adatot. Az EventService arra használja fel az adatot, hogy ellenőrzs alá vesse a létrehozandó eseményhez kiválasztott sportot, vajon létezik-e? A SportService viszont arra használja fel ugyanazt az eredményt, hogy egy prezentálható formában vissza téritse a Presentation szintnek a sportot, ami egy Controller formájában elérhetővé teszi a kliens számára, hogy lekérhessen egy sportot ID alapján.

### 3.1.2. Az adatbázis

A rendszer adatai egy Microsoft SQL Server relációs adatbázisban[17] vannak eltárolva. Mivel a rendszer személyes információkat tárol el a felhasználóról, szükséges volt egy olyan adatbázist választani, mely megfelelő képpen tudja kezelni az egyes felhasználók és felhasználókhöz kötődő entitások közötti kapcsolatokat. Továbbá az is fontos, hogy biztonságosan legyenek az adatok tárolva. Mindezeket figyelembe véve egy relációs adatbázis volt a legjobb választás. A konkrét döntés azért esett a Microsoft SQL Server termékre, mivel a szerver fő keretrendszerére alapból Microsoft termék, így a Microsoft-készített adatbázis egy jó integrálhatósági lehetőségnek bizonyul. Továbbá a technológiák megegyezése elősegít a későbbi felhőbe való telpítésben is.

Az adatbázis strukturális létrehozása a *code first approach*[3] elv segítségével hajtódik végre, ahogyan a 3.4-es ábrán is látható. Az Entity Framework Core lehetővé teszi azt, hogy adatmodellek és adatmodellek közti kapcsolatok alapján migrációk segítségével változtassunk az adatbázis sémán, illetve az abba tartozó adatokon.



3.4. ábra. A Code First Approach áttekintése

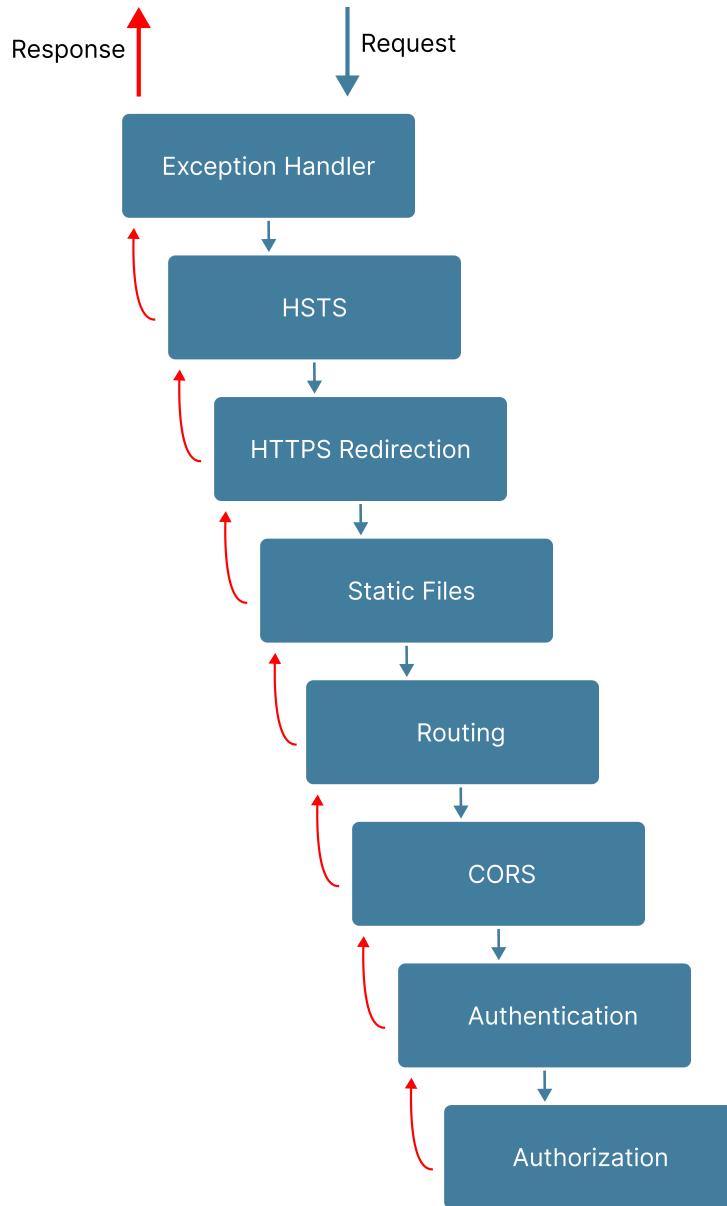
Technikailag megvizsgálva, a Code First Approach [3.4-es ábra] az Entity Framework Core egyik hozománya, mely az ORM (Object Relational Mapper)[3] segítségével képes a *DbContext* osztályon keresztül kapcsolatba lépni az adatbázissal, továbbá a *RepositoryBase* absztrakt osztályon végrehajtandó CRUD műveletek[4] segítségével tudja az itt tárolt entitásokat kezelni, törlni vagy újakat létrehozni.

### 3.1.3. A Cronotus API

A kliens oldali kéréseket a szerver a Presentation [3.1-es ábra] rétege fogadja és szolgálja ki. A *pipeline*-ja HTTPS (HyperText Transfer Protocol Secure)[7] protokolt követve üzemel. A kliens HTTP kéréseket küldve tud adatokat küldeni, lekérni, illetve szerveri erőforrásokat igénybe venni adatokkal kapcsolatos műveletekhez. Az adatokat DTO-k (Data Transfer Object)[5] formájában cserélgeti a kliens a szerver oldallal, mely JSON (JavaScript Object Notation)[10] formájában valósul meg.

A szerver minden beérkező kérést átfuttat egy ellenőrzésrendszeren [3.5-ös ábra], mely az Application Pipeline formájában van felépítve. Itt biztonsági és adatintegritási szempontból fontos vizsgálat alá veti a kéréseket. A szerver a pipeline-ban érvényesíti azt, hogy HTTP helyett biztosan HTTPS protokollon keresztül történjen meg a kommunikáció, hiszen így egy harmadik fél nem tudja leolvasni a küldött kérések tartalmát, úgy sem ha hozzáférése van ezekhez, mivel kódolva vannak. A pipeline továbbá más hasznos beállításokat is érvényesít, például azt, hogy csak megfelelő URL címekről legyen engedélyezett

a hozzáférés a szerverhez, illetve jogosultságot vizsgál *Authentication* és *Authorization*[1] segítségével.



**3.5. ábra.** A pipeline, melyen keresztül kell mennie minden szerver fele érkező kérésnek

### 3.1.4. Hitelesítés és Engedélyezés

Az rendszer JWT Tokenek[11] formájában kezeli a felhasználói munkamenetet és az ezekbe kódolt információk segítségével végzi el a felhasználók hitelesítését és bírálja el az *endpoint*-okhoz való hozzáférésüket.

Egy ilyen token fontos információkat tartalmaz, például felhasználói jogok, jogokhoz kapcsolódó id-k, illetve a token lejáratú dátuma.

A kliens oldalról kapott kérések mindegyikéhez csatolva van egy ilyen JWT token, amit a rendszer egy kérés beérkeztekor dekódol, majd az innen kinyert információk segítségével megnézi, hogy érvényes-e a token, majd ellenőrzi, hogy a felhasználónak van-e jogai végre hajtani a kívánt műveletet. Egy token élettartama 10 perc, ezután egy *Refresh Token* segítségével a kliens újat kell kérjen. Ez a fajta viszonylag rövid szavatossági idő egy fontos biztonsági intézkedés, mivel ha egy harmadik fél meg is tud szerezni egy ilyen érvényes tokent, akkor is csak maximum 10 perce van arra, hogy hozzáférése legyen a szerverhez.

```
public async Task<TokenDto> CreateToken(bool populateExp)
{
    var signingCredentials = GetSigningCredentials();
    var claims = await GetClaims();
    var tokenOptions = GenerateTokenOptions(signingCredentials,
        claims);

    var refreshToken = GenerateRefreshToken();
    _user!.RefreshToken = refreshToken;

    if (populateExp)
    {
        _user!.RefreshTokenExpiryTime = DateTime.Now.AddDays(7);
    }

    await _userManager.UpdateAsync(_user);
    var accessToken = new
        JwtSecurityTokenHandler().WriteToken(tokenOptions);
    return new TokenDto(accessToken, refreshToken);
}
```

### 3.1. kódrészlet. Egy JWT token felépítése a szerver oldalon

A 3.1-es kódrészlet bemutatja azt a szerver oldali függvényt, amely segítségével létre lehet hozni egy új JWT tokent. A *GetSigningCredentials* függvény egy byte sorozatot térít vissza, ez adja meg a tokenhez szükséges privát kulcsot. A *GetClaims* függvény segítségével előkészítjük azokat az információkat, melyeket bele kívánunk helyezni a jelenleg létrehozandó tokenbe. Ide tartoznak a fekhasználói jogok, a felhasználói jogokhoz tartozó egyedi azonosítók, továbbá a token lejáratú dátuma, melyet egy úgynevezett „Unix timestamp” formájában tárol a token. Továbbá észre vehető az is, hogy ugyanitt létrehozásra kerül egy *refresh token* is, mely egy 7 napos lejáratú idővel rendelkezik. Ez gyakorlatilag azt jelenti, hogy a felhasználó 7 napig tudja használni a webalkalmazást egyetlen bejelentkezéssel mindamellett, hogy az *access token* folyamatosan frissül a lehető legjobb biztonság érdekében. A függvény végső részében az adatbázisba is beírásra kerül a refresh

token lejáratú időpontja, illetve a létrehozott access és refresh token egy DTO segítségével visszatérítésre kerül.

### 3.1.5. Globális hibakezelés

Annak érdekében, hogy a kód átlátható és könnyen karbantartható legyen a hiba és kivételkezelést a projekt egy globális hibakezelő middleware segítségével oldja meg, mely részét képezi az application pipeline-nak.

Ha ezt a kezelési módszert használjuk, akkor nincs szükség *try - catch* kód blokkok használatára, hiszen ha valamilyen kivétel merül fel, akkor azt elég csak tovább dobni, majd a middleware automatikusan lekezeli. A kezelés során a middleware beállítja a visszatérítendő objektumba a fellépő hiba HTTP státuszkódját, majd a kiindulási pontból dobott hiba üzenetét, mindezt annak érdekében, hogy kliens oldalon könnyen olvasható legyen, hogy egyes hiba miből származhat.

## 3.2. A felhasználói felület

A felhasználói felület egy webalkalmazás formájában nyilvánul meg. A felhasználók itt végezhetnek eseményekkel kapcsolatos műveleteket, például létrehozás, csatlakozás, vagy kommentelés. Ez az alfejezet a felhasználói felület felépítését és működését tárgyalja.

### 3.2.1. Felhasznált technológia

A webalkalmazás TypeScript[22] nyelven íródott, React[15] könyvtár felhasználásával. A TypeScript a JavaScript[9] nyelvet bővíti ki egy szintaxis csomaggal, amely lehetővé teszi a változók és objektumok típuskezelését, ezzel egy jobban átlátható és biztonságosabb fejlesztési procedúrát nyújtva.

A React könyvtár segítségével a webalkalmazás egy Single Page Application (SPA)-ként[19] működik. A SPA az egész weboldalt egyetlen HTML dokumentumban építi fel, majd változás során ennek a tartalmát frissíti és cseréli. Ez előnyös megoldás, mivel így nem kell minden oldalcsere, illetve tartalomfrissítés után újratölteni az alkalmazást, így gyorsabb lesz, javítja a felhasználói élményt.

Az alkalmazáshoz felhasznált további könyvtárcsomagok kezelése, a webalkalmazás futtatása, illetve építése a Vite[23] segítségével van megoldva. A Vite egy gyors és egyszerű build eszköz, mely alkalmas React webalkalmazások kezelésére. Lehetővé tesz Live Reloading-ot mely hasznos a fejlesztés során, illetve úgy van felépítve, hogy minimalizálja az alkalmazás építéséhez szükséges időt, így még jobban gyorsítva és javítva a fejlesztési időt, élményt.

Az alkalmazás felhasználói felületének egy szerves része továbbá a React könyvtárra épített Material UI komponens csomag, mely segítségével könnyen és gyorsan lehet előre megépített megjelenítő komponenseket felhasználni.

### 3.2.2. Kommunikáció a szerverrel

A webalkalmazás HTTPS protokollon keresztül kommunikál a szerverrel. A HTTPS protokoll a HTTP protokoll biztonságos formája. Itt a HTTP-vel ellentétben az adatok nem egyszerű szöveges formátumban vannak küldve, hanem kódolt blokkokként, melyek biztosítják azt, hogy út közben egy rosszindulatú harmadik félnek ne legyen hozzáférése bármilyen kommunikációhoz.

Az alkalmazás GET metódusokra az SWR - Data Fetching[21] könyvtárcsomagot használja. Az SWR rövidítés az angol „stale-while-revalidate” szókapcsolatból ered, mely annyit tesz, hogy „elavult-közben-újraérvényesít”. Az SWR adatlekérési stratégia alapján egy kérés után először a potenciális elavult adatot megkapjuk egy cache (stale) tároló rendszerből, majd elküldjük a kérést a legfrissebb adatért (revalidate), majd végül előálunk ezzel.

A könyvtárcsomag lehetőséget biztosít arra is, hogy az adatlekéréssel kapcsolatos státuszokat kezeljük, például kérés alatt töltés ideje, illetve hibákat. Ezt egyszerűen megtehetjük a függvényhívás esetén, amint a 3.2-es kódrészleten is látszik.

---

```

const { data: profileInfo, isLoading, error, mutate } = useProfile(userId!);

{isLoading ? (
    <ProfileTitleIsLoading />
) : (
    <ProfileHead
        profileInformation={profileInfo!}
        className="profile-head"
    />
)}

```

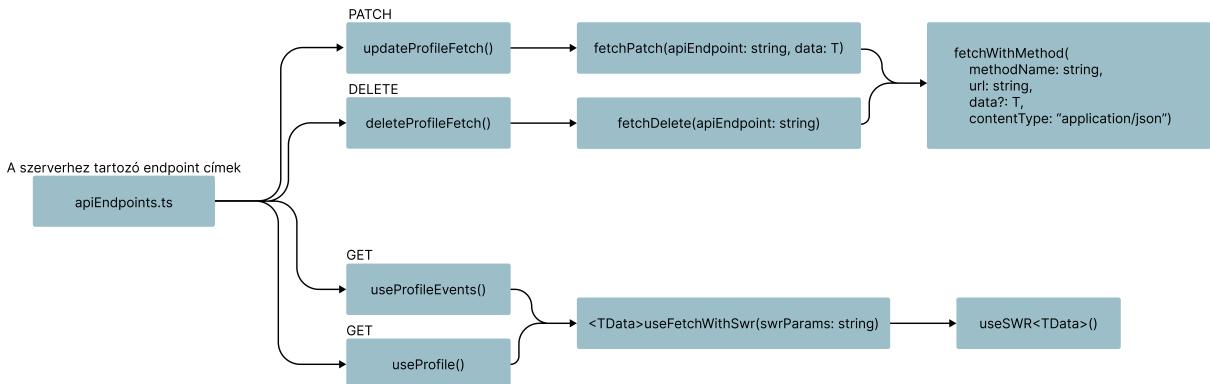
---

### 3.2. kódrészlet. SWR használata profil adatok lekérésére

A 3.2-es ábrán látható, hogy abban az esetben, ha az adatok még lekérés alatt vannak, a komponens a „ProfileTitleIsLoading” komponensemset tölti be, majd amint a a betöltés befejeződik, megjeleníti a betöltött adatokat.

Amennyiben a szerverhez intézett HTTP kérés nem GET metódus, úgy az alkalmazás nem SWR-t használ, hanem egy egyszerű, dinamikusan felépített fetchert. Ez a fetcher függvény újrahasznosítható, így elég minden egyedi kérés esetében csak megadni a megfelelő adatokat, endpoint címet, és meghívni az ehhez tartozó fetchert.

A 3.6-os ábra szemlélteti néhány végpont egyszerűsített működését.



3.6. ábra. Néhány endpoint működése

### 3.2.3. Oldalak közötti navigálás

A webalkalmazáson való oldalak közötti navigációért a React Router[16] könyvtár-csomag felelős. Ez lehetővé teszi azt, hogy könnyen és zökkenőmentesen tudunk oldalak között navigálni, illetve ennek során információt cserélni egyes oldalak, komponensek között.

A React Router lehetővé teszi azt, hogy deklaráljunk minden létező oldalt egy „Router” komponensben, ahonnan aztán a Single Page Application dinamikusan cserélni tudja

a pillanatnyilag megjelenített oldalt. Ez a deklarálási módszer látható a 3.3-as kódrészletben.

```
const Router = (props: { onLogout: () => void }) => {
  return (
    <Routes>
      <Route path="/" element={<Layout />}>
        <Route index element={<Home />} />
        <Route path="browser" element={<Browser />} />
        <Route path="create-event" element={<CreateEvent />} />
        <Route path="event/:id" element={<SportsEvent />} />
        <Route path="profile" element={<Profile onLogout={props.onLogout} />} />
      </Route>
      <Route path="*" element={<NoMatch />} />
    </Routes>
  );
};
```

### 3.3. kódrészlet. Route-ok deklarálása

Abban az esetben, ha kliens oldalon böngészőn belül nem a megfelelő cím kerül megadásra, úgy a „NoMatch” route kerül betöltésre. Ez hasznos, mivel ha a felhasználó egy nem létező oldalt akar betölteni, akkor nem egy érthetetlen hibaüzenet kerül elé, hanem egy általunk felépített érthető visszajelzés.

### 3.2.4. Munkamenetkezelés JWT tokenekkel

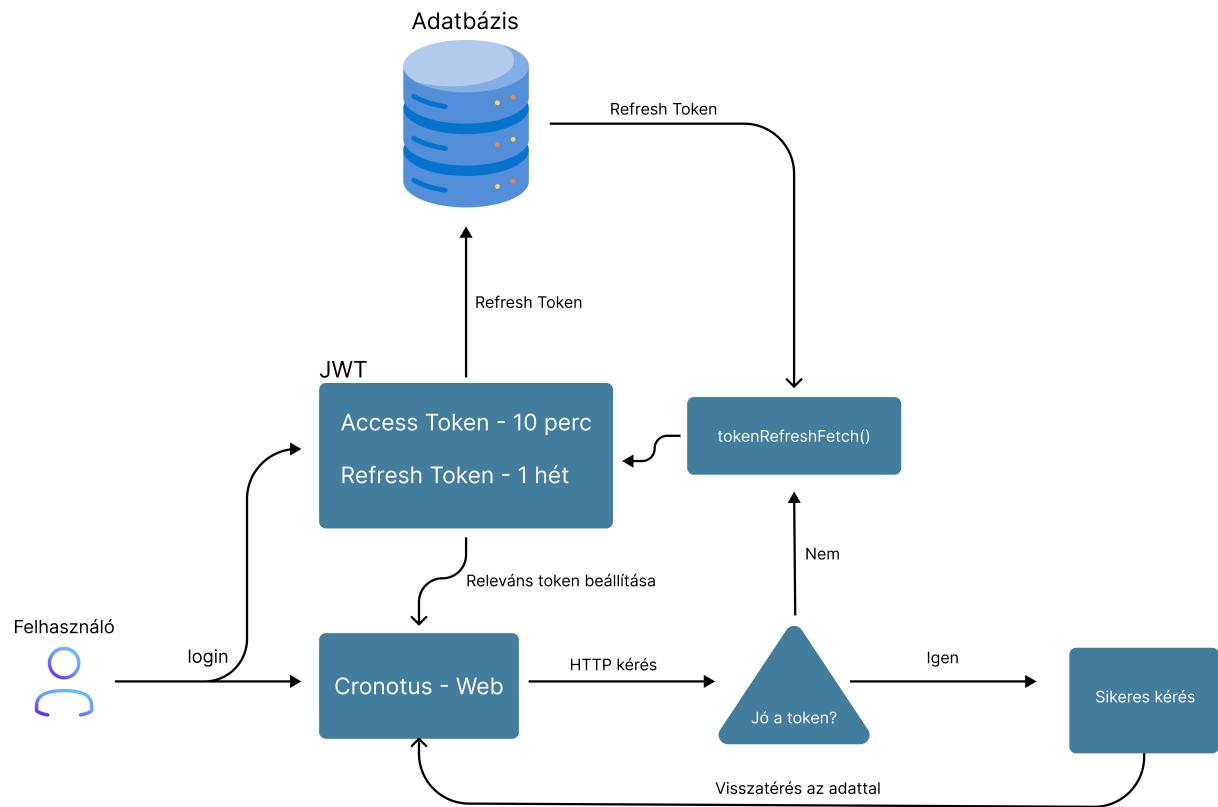
A webalkalmazás biztonságos működését JWT (JSON Web Token)-ek segítik elő. A JWT egy szerkezetileg három részből álló metódusa annak, hogy két fél biztonságosan információt válthasson egymás között.

A webalkalmazás szintjén ahhoz, hogy a Authentication és Authorization-el kapcsolatos feladatokat elvégezzük ilyen tokeneket használunk. minden szerverhez intézett kéréshez a HTTP kérés Header részébe egy ilyen token van csatolva, amely segítségével elérhetők a jogrendszer szerinti védett endpointok. A tokenekbe bele van kódolva a felhasználókhöz kapcsolódó egyedi azonosítók, jogok, illetve a token lejárási ideje.

A lejárási idő a webalkalmazás szintjén fontos, mivel ennek függvényében lehet a munkamenetet követni. Abban az esetben ha egy token lejár, a felhasználó alap esetben arra van kérve, hogy jelentkezzen be újra. Ez azért hasznos, mert ha egy harmadik rosszindulatú fél meg is szerez egy ilyen tokent, csak limitált ideje van kárt tenni a rendszerben, esetünkben 10 perc.

Ahhoz, hogy egy jó felhasználói élményt nyújtsan a webalkalmazás, két féle tokent használunk. Van access token, illetve refresh token. Az access tokenek magukba foglalnak minden olyan információt és működési elvet, melyet a fentiekben tárgyaltunk. Azért, hogy egy access token lejártakor ne kelljen újra bejelentkeznie a felhasználóknak, refresh tokeneket használ az alkalmazás, melyek adatbázisban tárolva vannak minden felhasználó számára. Ezek segítségével új access tokent lehet generálni, így nem szükségeltetve az újból bejelentkezést 10 percenként.

A [3.7](#)-es ábra bemutatja a JWT tokenek segítségével vezetett munkamanetkezelés logikáját.



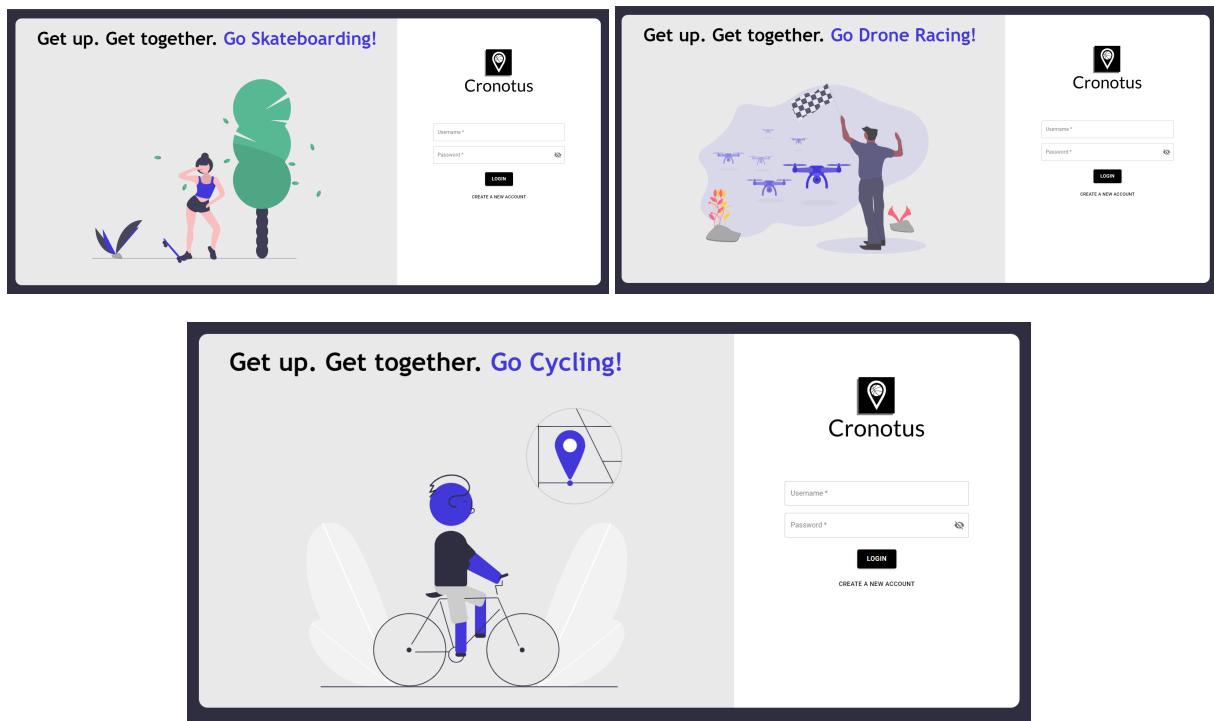
**3.7. ábra.** A JWT munkamenetkezelés folyamata

## 4. fejezet

# A Cronotus felhasználói felülete

### 4.1. A fogadó oldal és a regisztráció

A webalkalmazásra való látogatás során a felhasználókat egy egyszerű felület fogadja, ahol lehetőségük van új felhasználó regisztrálására, vagy belépésre egy már meglévő felhasználóval. Ez látható a 4.1-es ábrán.



4.1. ábra. A fogadó oldal

A webalkalmazás igyekszik egy játékos és barátságos hangnemben megnyilvánulni a felhasználók számára. Ez megfigyelhető a regisztrációs folyamatban is, ami nem egy megszokott űrlap kitöltéséből áll, hanem egyszerű nyelvezettel megfogalmazott kérdések sorozatával segíti a felhasználókat a regisztráció során. **4.2-es ábra**

What's your real name?

First name: Sándor Last name: Adám

NEXT

How should we call you?

Sándor Adám

NEXT

We'll need your email address too!

Email: sanderadam@gmail.com

NEXT

Let's create a strong password for you!

Password: Confirm password:

NEXT

At last, we need your phone number!

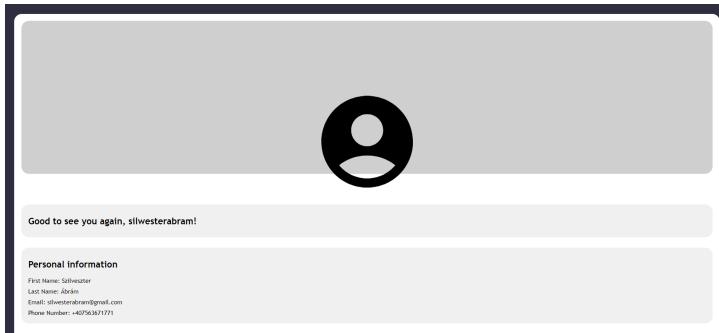
Phone number: +36705237177

NEXT

**4.2. ábra.** A regisztrációs folyamat

## 4.2. A profil oldal

A regisztráció alkalmával megadott információk megtekinthetők a profil oldalon. Itt lehetőség van arra is, hogy a felhasználó megváltoztassa a személyes információit, illetve profil és borítóképét is. Ez látható a 4.3-as ábrán.



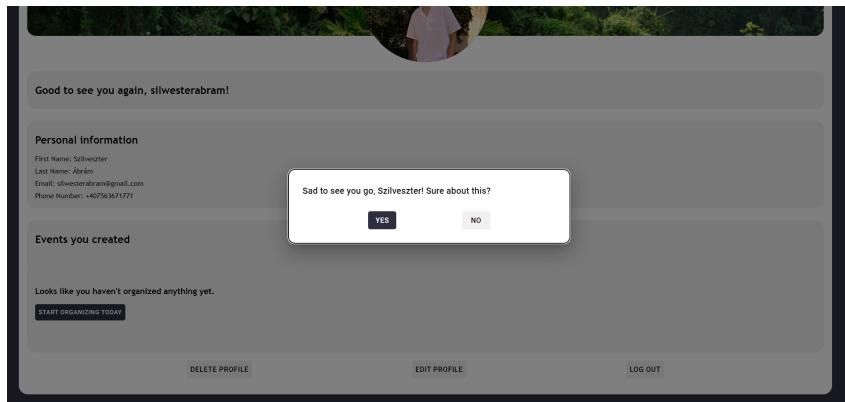
4.3. ábra. A profil oldal

A Profil oldalon való „Edit Profile”-ra való kattintással előhozható egy felület, ahol a felhasználó megváltoztathatja személyes információit, illetve lehetősége van arra is, hogy profil és borítóképet változtasson, hogy egyéni stílusát tükrözze a felületen. Ez látható a 4.4-es ábrán.

The image contains two screenshots of a web application. The top screenshot shows the "Edit profile" form. It has input fields for First name (Silvester), Last name (Ábra), Email (silvesterabram@gmail.com), and Phone number (+407563671771). Below the form are four buttons: "SAVE" (disabled), "CANCEL", "NEW COVER IMAGE", "NEW PROFILE PICTURE", "DELETE COVER IMAGE", and "DELETE PROFILE PICTURE". At the bottom are links for "DELETE PROFILE", "EDIT PROFILE", and "LOG OUT". The bottom screenshot shows the user's profile page after saving changes. It features a large, circular profile picture of a man standing in front of a waterfall. Above the picture is the message "Good to see you again, silvesterabram!". Below the picture is the "Personal Information" section with the same details as the first screenshot. At the very bottom are links for "Home", "Browser", and "Profile".

4.4. ábra. A profil szerkesztése

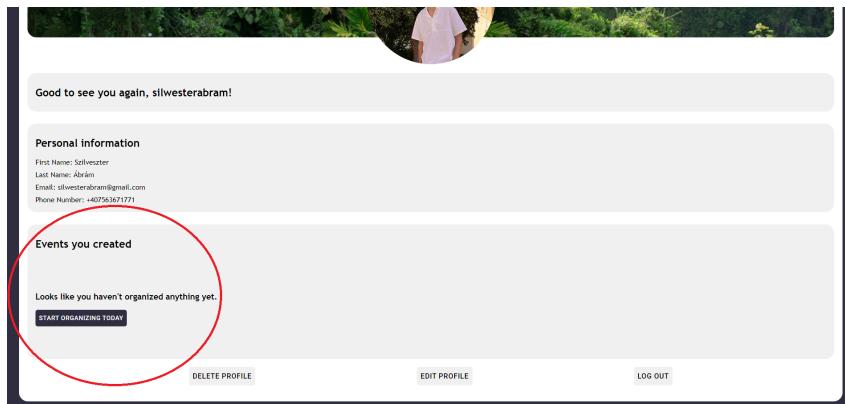
A továbbiakban lehetőségünk van a felhasználónk törlésére is. A „Delete Profile” gomb megnyomásával előhozható egy prompt, amit ha továbbra is megerősítünk, akkor



**4.5. ábra.** A profil törlése

egyszerűen törölhetjük a felhasználónkat, s minden vele kapcsolatos információt, eseményt. [4.5-ös ábra]

Az oldal listázza az általunk szervezett sporteseményeket is. Mivel egy újonnan létrehozott felhasználónk van, még nem hoztunk létre egyetlen sporteseményt sem, így a lista üres. Ilyenkor ezt jelzi nekünk az oldal, illetve ennek megfelelően jelenít meg egy gombot, amivel átirányít bennünket arra a felületre, ahol létrehozhatunk új sporteseményeket. [4.6-os ábra]



**4.6. ábra.** Általunk szervezett sportesemények listázása a profil oldalon

### 4.3. A sportesemény böngésző

A sportesemény böngésző, azaz a „Browser” oldal két fő részre osztható. A „See what others are up to” fül alatt listázhatsuk az összes létrehozott sporteseményt, míg a „Create a new event” lehetőséget ad arra, hogy létrehozhassuk egy új, saját eseményt. Ez látható a 4.7-es ábrán is.



**4.7. ábra.** A sportesemény böngésző

Egy új esemény létrehozása során egy egyszerű űrlapot kell kitöltenie a felhasználónak. Itt meglévő sportok közül választhat, az esemény témajaként. Megfelelő információk, sport, dátum és résztvevőszám megadása után a „Submit” gombra kattintás hozhatja létre az általa kívánt eseményt. Ez a folyamat látható a 4.8-as ábrán. Az esemény sikeres létrehozása után megfigyelhetjük, hogy az esemény megjelenik a sportesemény böngészőben is. Itt rákattintás segítségével előhozható egy részletesebb nézet. [4.9-es ábra]

The image consists of four screenshots illustrating the process of creating a new sport event. The screenshots are arranged in a 2x2 grid. All screenshots have a header with 'Home', 'Browser', and 'Profile' tabs, and a 'Create a new event' button in the top right corner.

- Screenshot 1 (Top Left):** Shows the initial 'Let's create your own event!' form. It includes fields for 'Event name' (set to 'Football & Friends'), 'Sport' (set to 'Football'), and 'Start Date' (set to '04/16/2024 07:44 AM'). Below the form are 'Submit' and 'Back' buttons.
- Screenshot 2 (Top Right):** Shows a list of available sports: American Football, Badminton, Basketball, Cricket, Field Hockey, Football, Frisbee, Handball, Ice Hockey, Lacrosse, Rugby, Softball, Table Tennis, Tennis, Volleyball, Water Polo.
- Screenshot 3 (Bottom Left):** Shows the date selection interface for setting the start date. A calendar for April 2024 is displayed, with the date '04/16/2024' selected. Below the calendar are 'Submit' and 'Back' buttons.
- Screenshot 4 (Bottom Right):** Shows the completed event creation form with all fields filled: Event name ('Football & Friends'), Sport ('Football'), Start Date ('04/16/2024 07:44 PM'), and Guests ('0'). Below the form are 'Submit' and 'Back' buttons.

**4.8. ábra.** Esemény létrehozása



**4.9. ábra.** Az újonnan létrehozott esemény a sportesemény böngészőben

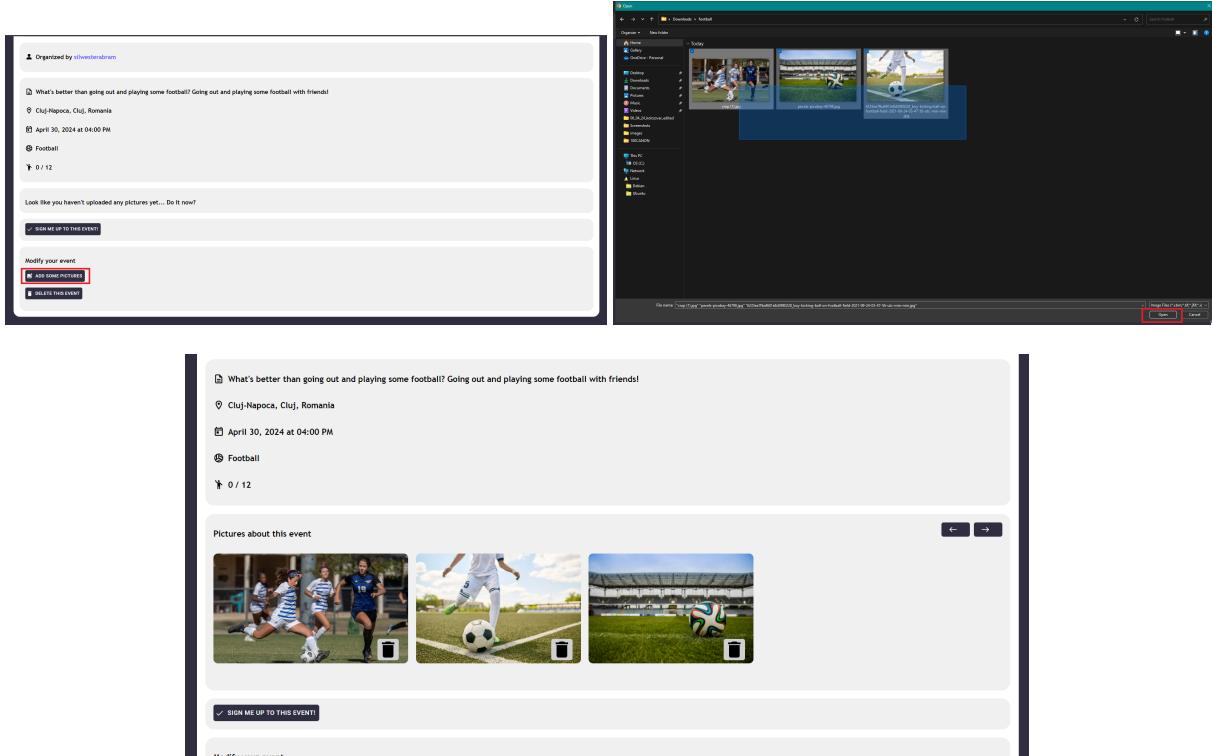
Az esemény részletes megjelenítésében látható minden általunk megadott információt. Az esemény azt is mutatja, hogy ki az esemény szervezője. A szerző nevére kattintva megtekinthető egy rövid leírás a szervező profiljáról. Ez látható a 4.10-es ábrán is.

**4.10. ábra.** Az esemény szervezőjének megjelenítése

Egy esemény létrehozásakor az esemény szervezője automatikusan feliratkozik erre. Amennyiben ezt vissza szeretné vonni, a „Resign me from this event” gomb segítségével megteheti. Ilyenkor megfigyelhető, hogy az eseményre jelentkezett egyének száma egyel csökken. Ez megfigyelhető a 4.11-es ábrán.

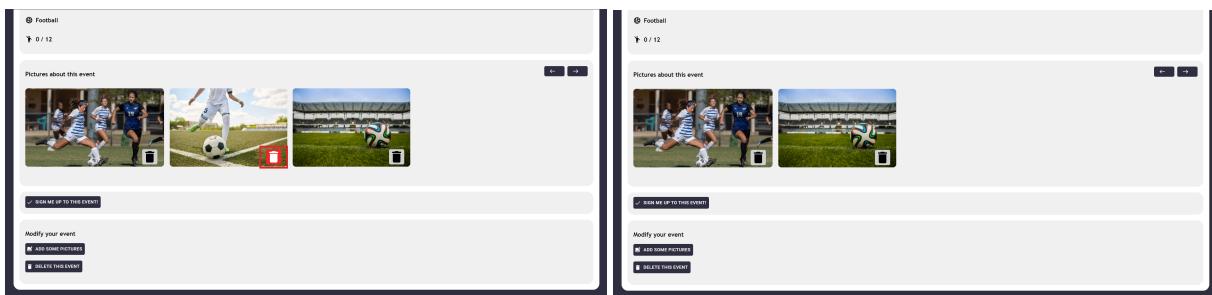
**4.11. ábra.** Egy eseményről való leiratkozás

Szervezőként adhatunk hozzá képeket is az általunk szervezett eseményekhez. Ezt az „Add some pictures” gombra való kattintással tehetjük meg. Ez felhoz egy ablakot, majd itt a képek kiválasztása után az oldal frissül és láthatóak lesznek az általunk kiválasztott képek. Ez a folyamat látható a 4.12-es ábrán.



**4.12. ábra.** Képek hozzáadása az eseményhez

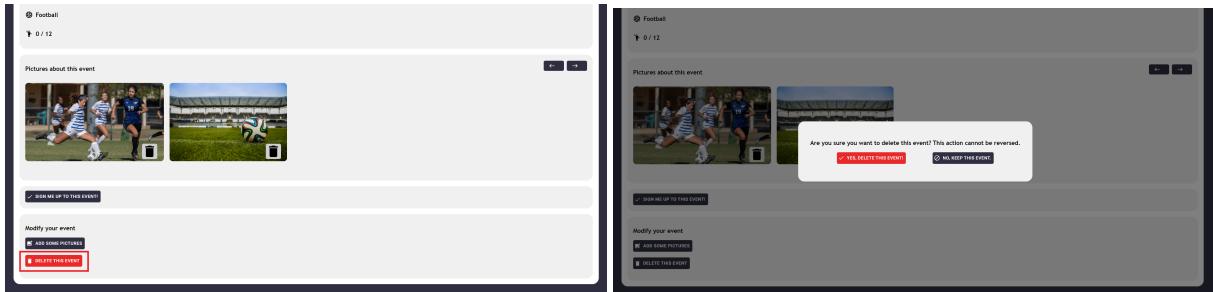
Egy esemény szervezője törlhet is képeket az eseményről. Ezt a kép jobb alsó sar-kában megjelenő törlés ikonra kattintással teheti meg. Ilyenkor az oldal tartalma automatikusan frissül, s a törlni kívánt kép eltűnik az esemény oldaláról. Ez megfigyelhető a 4.13-as ábrán.



**4.13. ábra.** Kép törlése az eseményről

Végső soron a szervező törlheti is az általa szervezett eseményt. Ezt a „Delete this event” gombra való kattintással teheti meg, ahogyan a 4.14-es ábrán is látható. Ilyen-

kor egy megerősítendő ablak jelenik meg, amiben ha a „Yes, delete this event!” gombra kattintunk, az esemény véglegesen törlése kerül. Ezután az oldal átirányít bennünket a böngésző felületre, ahol már nem fogjuk látni az általunk törölt eseményt.

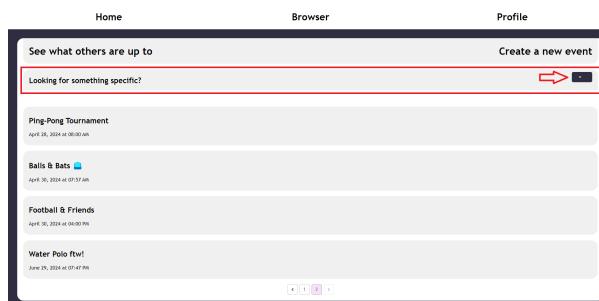


**4.14. ábra.** Esemény törlése

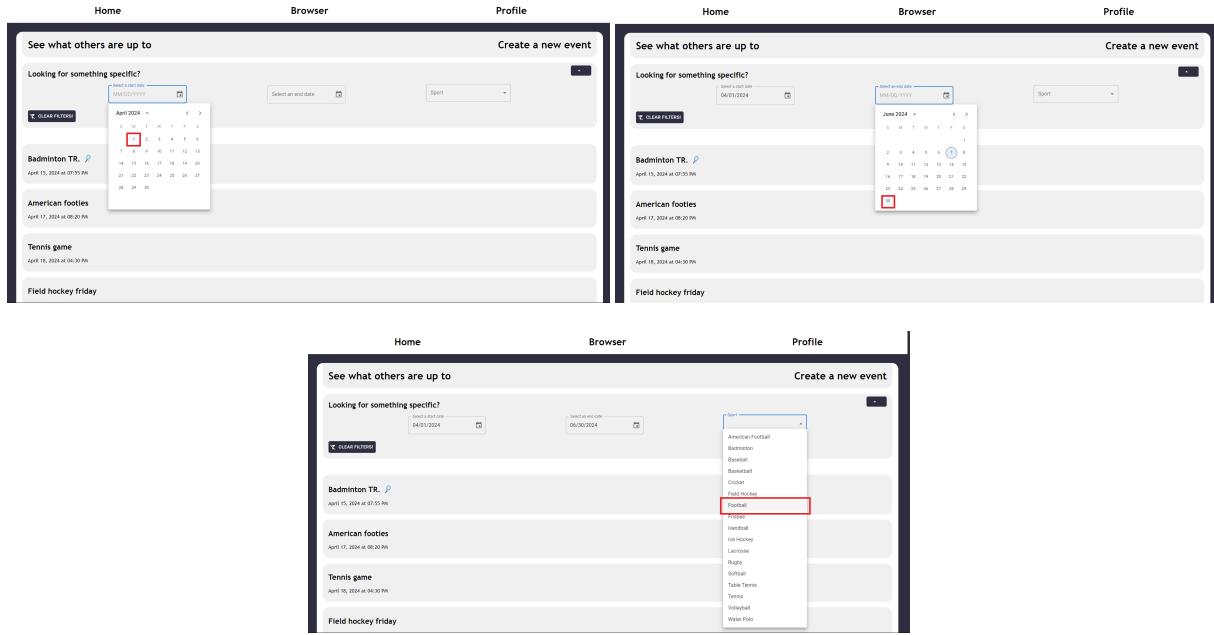
### 4.3.1. Sportesemények szűrése

A sportesemények böngészése során a felhasználónak lehetősége van különböző szűrők használatára. A szűrők segítségével meg lehet határozni egy időintervallumot, amely alatt az eseményeket meg szeretnénk tekinteni, illetve kiválasztható egy specifikus sport is, amely alapján lehetősége van a felhasználóknak szűrni az eseményeket.

A szűrők használatához először kattintsunk a „Looking for something specific?” fülre:



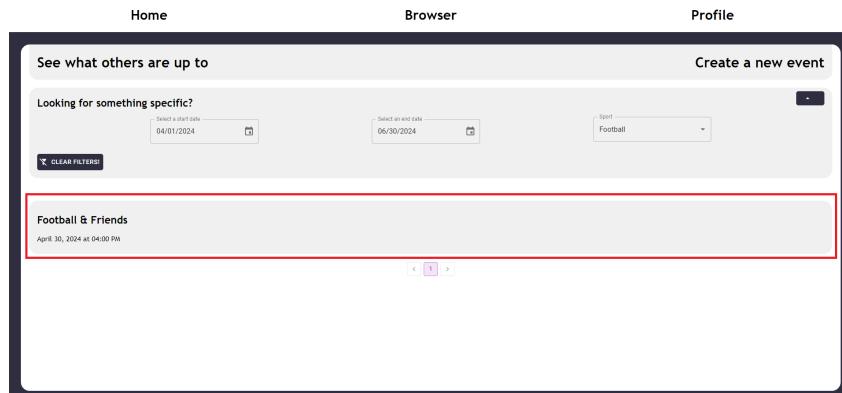
**4.15. ábra.** A szűrőket tartalmazó fül előhozása



**4.16. ábra.** A szűrők beállítása

Amint az látható a 4.16-es ábrán, a szűrők beállítása után az oldal automatikusan frissül, és a beállított szűrőknek megfelelően jeleníti meg az eseményeket.

Ahogyan a 4.17-es ábrán is látható, a szűrők alkalmazása egyetlen sporteseményt eredményezett.



**4.17. ábra.** A szűrt események

## 4.4. Lapszámozás - Pagination

Annak érdekében, hogy jó teljesítménnyel működhessen a webalkalmazás, szükséges az ú.n. *Pagination*[14], azaz az adatbázisból lekért sportesemények szeletekre osztása. Ez azt jelenti, hogy nem fog minden felhasználó minden létező eseményt megkapni egy kérés során, hanem csak egy limitált számot, melyek szekvenciális továbbá lekérhetőek.

Az alkalmazás automatikusan 5-ös szeletekre osztja a sporteseményeket, ezek között a lap alján van lehetősége a felhasználónak a navigálásra. Ez látható a 4.18-os ábrán.



4.18. ábra. A lapozás

Itt követehető az aktuális lap száma, illetve lehetőség van tovább haladásra a következő lapra, amennyiben ez létezik.

## 5. fejezet

# A projekt menedzselése

Annak érdekében, hogy a Cronotus projekt fejlesztési fázisa célra vezető és zökkenőmentes legyen, hasznos különféle projektmenedzselési eszközöket felhasználni.

### 5.1. Verziókezelés Git segítségével

A Cronotus projekt verziókezelése Git[24] segítségével történt. A Git azért volt egy hasznos eszköz a projekt keretein belül, mivel így könnyen számon lehet tartani az egyes verziók közötti különbséget, illetve ha az újonnan beiktatott kód nem megfelelő, könnyen el lehet érni korábbi, működő verziókat. A projekt verziókezelésére külön git branch-ek voltak használva, így minden nagyobb, elkülöníthető funkció egy külön ágat kapott a jobb átláthatóság érdekében.

A forráskód publikusan elérhető a <https://www.github.com/cronotus> weblapon.

Itt a projekt kliens oldali kódja megtalálható a <https://www.github.com/cronotus/frontend>, míg a szerver oldali kódja a <https://www.github.com/cronotus/backend> címen érhető el.

# Továbbfejlesztési lehetőségek és következtetések

A Cronotus projekt jelenlegi állapotában tartalmaz minden olyan alap funkciót, mely szükséges az alapszintű működéshez és a cél eléréséhez, viszont számos továbbfejlesztési lehetőség rejlik benne, melyekkel a felhasználói élményt tovább lehetne javítani, illetve a funkcionalitást bővíteni:

- **Email alapú értesítések:** Email alapú értesítések bevezetése, melyek a felhasználókat értesítenék az eseménykről, melyek érdeklik őket, vagy azokról, melyekben részt vesznek.
- **Google térkép integráció:** A Google térkép integrációja, mely lehetővé tenné a felhasználók számára, hogy megtekinthessék az események helyszínét a térképen.
- **Visszajelzés rendszer bevezetése:** Egy visszajelzés rendszer bevezetése, mely lehetővé tenné a felhasználók számára, hogy értékeljék az eseményeket, illetve a szervezőket, és ezáltal segítsék a többi felhasználót a döntésben.
- **Barátok:** Egy rendszer, ami lehető teszi azt, hogy felhasználókat jelölhessünk be barátként, és így könnyebben megtalálhassuk azokat az eseményeket, melyek érdekelhetik őket.
- **Statisztikai kimutatások:** Időszakos statisztikai kimutatások, amelyek jelzik, hogy egy adott időintervallumban mely sportokat végzett egy felhasználó aktívan, illetve milyen eseményeken vett részt.

# Irodalomjegyzék

- [1] Authentikációs folyamatok - auth0 dokumentáció. <https://auth0.com/docs/flows>. [Hozzáférés ideje: 2024. 04. 14].
- [2] C# hivatalos dokumentáció. <https://docs.microsoft.com/en-us/dotnet/csharp/>. [Hozzáférés ideje: 2024. 04. 14].
- [3] Code first approach - microsoft dokumentáció. <https://docs.microsoft.com/en-us/ef/ef6/modeling/code-first/workflows/new-database>. [Hozzáférés ideje: 2024. 04. 14].
- [4] Crud műveletek - microsoft áttekintés. <https://learn.microsoft.com/en-us/iis-administration/api/crud>. [Hozzáférés ideje: 2024. 04. 14].
- [5] Dto - microsoft dokumentáció. <https://learn.microsoft.com/en-us/aspnet/web-api/overview/data/using-web-api-with-entity-framework/part-5>. [Hozzáférés ideje: 2024. 04. 14].
- [6] Entity framework hivatalos dokumentáció. <https://docs.microsoft.com/en-us/ef/>. [Hozzáférés ideje: 2024. 04. 14].
- [7] Http - mozilla dokumentáció. <https://developer.mozilla.org/en-US/docs/Web/HTTP>. [Hozzáférés ideje: 2024. 04. 14].
- [8] Inversion of control - microsoft dokumentáció. <https://docs.microsoft.com/en-us/aspnet/core/fundamentals/dependency-injection>. [Hozzáférés ideje: 2024. 04. 14].
- [9] Javascript - mozilla dokumentáció. <https://developer.mozilla.org/en-US/docs/Web/JavaScript>. [Hozzáférés ideje: 2024. 04. 14].
- [10] Json hivatalos dokumentáció. <https://www.json.org/json-en.html>. [Hozzáférés ideje: 2024. 04. 14].

- [11] Jwt - json web token áttekintés. <https://jwt.io/introduction/>. [Hozzáférés ideje: 2024. 04. 14].
- [12] .net hivatalos dokumentáció. <https://docs.microsoft.com/en-us/dotnet/>. [Hozzáférés ideje: 2024. 04. 14].
- [13] Onion architecture - microsoft dokumentáció. <https://docs.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/common-web-application-architectures>. [Hozzáférés ideje: 2024. 04. 14].
- [14] Pagination - techtarget áttekintés. <https://www.techtarget.com/whatis/definition/pagination>. [Hozzáférés ideje: 2024. 06. 07].
- [15] React hivatalos dokumentáció. <https://devdocs.io/react/>. [Hozzáférés ideje: 2024. 04. 14].
- [16] React router hivatalos dokumentáció. <https://reactrouter.com/>. [Hozzáférés ideje: 2024. 04. 14].
- [17] Relational database - digitalocean áttekintés. <https://docs.digitalocean.com/glossary/relational-db/>. [Hozzáférés ideje: 2024. 04. 14].
- [18] Restful api hivatalos dokumentáció. <https://restfulapi.net/>. [Hozzáférés ideje: 2024. 04. 14].
- [19] Single page application - microsoft dokumentáció. <https://learn.microsoft.com/en-us/entra/identity-platform/index-spa>. [Hozzáférés ideje: 2024. 04. 14].
- [20] Sql server hivatalos dokumentáció. <https://docs.microsoft.com/en-us/sql/>. [Hozzáférés ideje: 2024. 04. 14].
- [21] Swr hivatalos dokumentáció. <https://swr.vercel.app/>. [Hozzáférés ideje: 2024. 04. 14].
- [22] Typescript hivatalos dokumentáció. <https://www.typescriptlang.org/docs/>. [Hozzáférés ideje: 2024. 04. 14].
- [23] Vite hivatalos dokumentáció. <https://devdocs.io/vite/>. [Hozzáférés ideje: 2024. 04. 14].
- [24] Git hivatalos dokumentáció. <https://git-scm.com/doc>, [Hozzáférés ideje: 2024. 06. 21].

- [25] S. Warner, E. Sparvero, S. Shapiro, and A. Anderson. Yielding healthy community with sport? *Journal of Sport for Development*, 5(8):41–52, 2017.