Szakképesítés megnevezése: **Szoftverfejlesztő**

OKJ száma: **54 213 05**

ZÁRÓDOLGOZAT

Light Airlines – Android applikáció

Merényi Miklós

témavezetőNyírő Levente Gyula

14.S

Budapest, 2020

*Nyilatkozat*

*Alulírott Nyírő Levente Gyula kijelentem, hogy ez a záródolgozat saját tudásom, önálló munkám terméke.*

*Kijelentem, hogy a záródolgozat beköttetett és elektronikus formában leadott példányai mind formátumban, mind tartalomban egyezőek, eltérést nem tartalmaznak.*

*Dátum:*

*hallgató aláírása*

Tartalom

[1. Bevezetés2 4](#_Toc132911569)

[2. Témaválasztás **Hiba! A könyvjelző nem létezik.**](#_Toc132911570)

[3. Fejlesztői dokumentáció **Hiba! A könyvjelző nem létezik.**](#_Toc132911571)

[3.1. Az alkalmazott fejlesztői eszközök **Hiba! A könyvjelző nem létezik.**](#_Toc132911572)

[3.2. Adatmodell leírása **Hiba! A könyvjelző nem létezik.**](#_Toc132911573)

[3.3. Részletes feladatspecifikáció **Hiba! A könyvjelző nem létezik.**](#_Toc132911574)

[3.3.1. segedOsztaly package **Hiba! A könyvjelző nem létezik.**](#_Toc132911575)

[3.3.2. kezdoActivity package **Hiba! A könyvjelző nem létezik.**](#_Toc132911576)

[3.3.3. userActivity package **Hiba! A könyvjelző nem létezik.**](#_Toc132911577)

[3.3.4. adminActivity package **Hiba! A könyvjelző nem létezik.**](#_Toc132911578)

[3.3.5. globalActivity package **Hiba! A könyvjelző nem létezik.**](#_Toc132911579)

[3.4. Tesztelési dokumentáció **Hiba! A könyvjelző nem létezik.**](#_Toc132911580)

[3.5. Továbbfejlesztési lehetőségek **Hiba! A könyvjelző nem létezik.**](#_Toc132911581)

[4. Felhasználói dokumentáció **Hiba! A könyvjelző nem létezik.**](#_Toc132911582)

[4.1. A program általános specifikációja **Hiba! A könyvjelző nem létezik.**](#_Toc132911583)

[4.2. Rendszerkövetelmények **Hiba! A könyvjelző nem létezik.**](#_Toc132911584)

[4.3. A program telepítése és konfigurálása **Hiba! A könyvjelző nem létezik.**](#_Toc132911585)

[4.4. A program használata **Hiba! A könyvjelző nem létezik.**](#_Toc132911586)

[5. Összegzés **Hiba! A könyvjelző nem létezik.**](#_Toc132911587)

[Forrásjegyzék **Hiba! A könyvjelző nem létezik.**](#_Toc132911588)

[Ábrajegyzék **Hiba! A könyvjelző nem létezik.**](#_Toc132911589)

# Bevezetés

Hároméves programtervező informatikai képzésem utolsó félévében létre kellett hoznom egy szoftvert az általam választott témában. Mindennek a tervezési, illetve implementációs lépéseit kellett véghez vinni. Ezen dokumentáció keretei között részletesen meg lehet ismerkedni a témaválasztás okaival, a tervezésem során kitűzött célokkal, az adatbázis tervezésével, a fejlesztési lépésekkel, a munkafolyamattal, illetve a megvalósított program jellemzőivel. Bemutatásra kerül a felhasználói dokumentáció, amelyben a felület funkcióit tervezem bemutatni, utána a fejlesztői részre áttérve kiderül a program mögöttes tartalma minden egyes funkció implementációjával együtt.

Témakörként a full stack technológiára esett a választásom magában foglalva a szerver- és a kliensoldali programozást. A téma mindig is érdekelt az informatika keretein belül. Hatással volt rám az adatstruktúrák kialakítása, adatkezelések, frontend és a backend közötti kapcsolat, illetve a dizájn kialakítása. A jövőben is ezzel a témakörrel tervezek foglalkozni, így nem volt más választásom, minthogy egy ezzel kapcsolatos szoftvert valósítsak meg.

Egy bekezdés erejéig pillantsunk vissza az időben: weboldalak szerves részeit képezték az internetnek annak elindulásával és a terjedésével. Amikor a lakosság számára elérhetővé vált az internet, akkor még statikus weblapok léteztek. Magyarországon mindez az 1990-as évek elején kezdődött el. Ezt a korszakot WEB 1.0-nak említik. A lényege volt, hogy információkat szerezzünk egy-egy weboldalról, amit valaki előre megszerkesztett. Később már megjelentek a mögöttes adatbázisok, logikák, amelyben már a felhasználók is lehetőséget kaptak az adatmanipulációra. Szükség volt egy adatátviteli technológiára, amelyre először az XML formátumot használták, azóta a legfőbb technológia erre a JSON, ami kisebb helyigénnyel rendelkezik.

A szakdolgozat is ezt a témát öleli körül. Az adatbázis migrációjára, illetve a háttérben futó kódok implementációjára a .NET technológia által nyújtott lehetőségekre támaszkodtam, mindezt C# programozási nyelven létrehozva. Az adatbázis mind MySql-en, mind MSSql-en képes kezelni az adatokat. A kinézeti részhez az Angular keretrendszert használtam TypeScript nyelvben a weboldal dinamikus részeit kialakítva. Mindemellett szükség volt HTML, SCSS és Bootstrap ismeretére is.

A szakdolgozat célja egy olyan webapplikáció felépítése volt, amely az autóverseny bajnokságok pontozási rendszereit bonyolítja le minden kezelhető esettel együtt. Mindebben lehetőség nyílik a felhasználókezelésre, majd létező felhasználóval saját szezonok létrehozására, azokban adatmanipulációra, illetve statisztikák megtekintésére. Fontos volt a weboldal mögött megalkotni a program nem látható részét, azaz a hátsó logikát, minden controller-rel és service-el együtt, amely összekapcsolja az adatbázis részt a frontend felülettel.

A fejlesztés elején számtalan kitűzött cél volt. Tudtam, hogy ezt a technológiát meg szeretném tanulni olyan szinten, hogy később ebben rugalmasabban tudjak szoftvereket készíteni. Nagy hangsúlyt fektettem mind a strukturált programozásra, mind a clean coding-ra egyaránt, hogy a szoftver később is karbantartható legyen. Sok kutatás és utánajárás növelte az önbizalmamat a .NET terén, ugyanis ez egy viszonylag új technológia volt számomra egy ekkora projekt keretein belül. A szoftver kinézeti részénél is hasonlóan a magabiztosságra törekedtem, ahol szintén úgy érzem, hogy teljesítettem a magamtól elvárt szintet, ugyanis rengeteg hibára bukkantam fejlesztés közben, amelyek kisebb-nagyobb kutatómunkát igényeltek. Itt is cél volt, hogy az önbizalmam meglegyen később az ilyen és ehhez hasonló technológiákhoz (Vue, React).

## 1.1 Témaválasztás

Nem kellett sokat gondolkodnom a témán, ugyanis volt bennem néhány ötlet, amit az idők során terveztem megvalósítani és az egyik ilyen terv megvalósítására kiváló alkalom volt, hogy mindezt egy szakdolgozat keretei között tegyem meg.

Baráti körben évekkel ezelőtt elkezdtünk játszani autóversenyzős játékokkal. Ez egy idő után olyan szintre emelkedett, hogy saját ligákat szerveztünk újabb és újabb versenyzőket bevonva. Egy verseny végén mindig volt egy eredmény, de általában nem álltunk meg egy versenynél, úgyhogy szerveztünk mellé még egy egész szériát tele versenyekkel, külön versenypályákon, hogy az összesített pontszám számítson a végén. Ehhez persze valahol vezetni kellett a pontokat, hogy a végén megszülessen a végső eredmény, hogy megtudjuk a legügyesebb versenyző nevét. Régebben erre az Excel szolgált, de amikor elkezdtem a szoftvertervezés és fejlesztés irányába mozdulni, megígértem a többieknek, hogy csinálok erre egy szoftvert, hogy elég legyen csak felcsapni a weboldalt ahelyett, hogy egy mindig frissülő, nehezen áttekinthető Excel táblát nézegetnének. Tudtam, hogy ez egy komplex feladat lesz, ugyanis sokféle esetet kell kezelni. Egy-egy szezon külön táblát igényel, ebben kezelni kell a versenyek időpontjait, figyelni kell a csapatbeosztásokra, ki melyik csapatnál szerzett pontot melyik versenyen és hogy közben milyen pozíciót ért el. Ebben a webapplikációban nem csak ezekre kellett figyelni, hanem az áttekinthetőségre is, ugyanis ez adja a végső megoldást, hogy kiváltsuk az egy fokkal bonyolultabb Excel táblát és közben egy valós idejű képet adjon, nem úgy, mint egy chaten átküldött fájl.

# Felhasználói dokumentáció

A következő fejezet célja, hogy iránymutatást adjon az új felhasználónak az elkészített program használatára. Megtudhatja a sorrendet, hogy hogyan juthat el egy végleges liga elkészítéséig, azokon a lépéseken keresztül, mint az felhasználó létrehozása, bejelentkezés, szezon létrehozása, versenyek, csapatok, pilóták hozzáadása, pontozása, illetve egyes beállításokkal is megismerkedhet.

## A program általános specifikációja

Sokan, akik autóversenyeznek, szeretjük a pontjainkat összesíteni. Ezt megtesszük akár papíron, akár telefonon, legtöbbször Excelen. Ha mindezt többször megtesszük jó látni egy végső eredményt, akár statisztikát, hogy hogyan állunk eredmény tekintetében, ki hogyan szokott teljesíteni egy-egy versenyen és az ilyen számokat feltöltő dolog elemezni. Ebből derülhet ki, hogy hogyan is kéne teljesítenem, min kéne javítanom.

A kutatásaim során azt vettem észre, hogy ahány liga volt, mind más pontozási rendszert, illetve összesítést használt. Mindenki másképp értelmezte az Excel által nyújtott lehetőségeket és más táblát adott végeredményként, amire azt más bajnokságoknál nem is vártunk volna. Az egyik motivációja volt így a szakdolgozatomnak, hogy egy olyan egységes rendszert hozzak létre a pontok összesítésére, amelyet bármelyik liga tud használni és egyértelműsíthetjük, hogy melyik ábra mit jelenthet. Ezért is tartalmaz az alkalmazás több szezont egyszerre. Ebből a rendszerből ráadásul olyan adatokat is kinyerhetünk, amely visszaadja a versenyző átlagos teljesítményét. Ez a rendszer túlmutat azon, hogy egy versenyző egy ligába bekerül és csak az ottani eredményéről tudunk.

Az is a weboldal mellett szól, hogy dinamikus az adatok tekintetében. Új adatok hozzáadása, vagy meglévő adatok módosítása esetén az eddigi adatok változhatnak anélkül, hogy nekünk bármit is kéne változtatni. Ez kizárja a hibafaktorokat, félreszámolásokat, ami nagy mértékben megkönnyíti a feladatunkat.

* 1. Felhasznált módszerek

Egy weboldalnak mindig vannak általánosan megszokott elemei, amiktől nem szabad vagy legalábbis nem illik eltérni. Fontos volt a regisztrációs, illetve a bejelentkezés weboldal, ugyanis ezek az alap helyek, ahol az autentikációt kezelhetjük. Ez kulcsfontosságú, hogy a létrehozott szezonokat tudjuk kihez kötni. Meg fog jelenni később egy főoldal, ahol az összes szezont láthatjuk kilistázva, illetve azon belül megtaláljuk majd minden egyes szezon részleteit, amiket a felhasználó hozzáadott. Ez adja majd a programnak az egyik lényegi részét. A frontend alapjában véve komponensalapú, így minden komponensnek meg kellett oldani az adatátvitelét és a saját stílusdizájnját. Előnye, hogy minden olyan komponenst, ami többször szerepelt, elég volt csak egyszer implementálni.

A backend résznél fontos volt kialakítani a lehető legáttekinthetőbb adatstruktúrát, ahogy szintén figyelni kellett az adatbázis szerkezet kialakítására, illetve a felhasználó- és jelszókezelésre is. Ezek mind-mind kutatómunkát igényeltek, mert .NET keretrendszerben ebben még nem teljesen volt lehetőségem ilyet implementálni.

// meséljünk az authentikációról és a roleokról

## Oldalak

Ahhoz, hogy tudjuk kezelni ezt az egész webapplikációt, tisztában kell lennünk a minden egyes oldal és kattintás funkciójával. A következő fejezetekben ezt tervezem részletezni.

Kezdetben a mindenhol megjelenő fejléc (header) komponenssel kell kezdenem. Itt megtalálhatjuk az összes olyan menüpontot, ami egy általános, egyáltalán nem specifikus oldalt nyit meg. Amennyiben nem vagyunk még bejelentkezve, akkor itt a szezonok és a statisztikák hivatkozások látszódnak, mellette pedig egy kék bejelentkezés gomb. Amennyiben létrehoztuk a profilt és már regisztráltunk is, akkor megjelenik egy harmadik hivatkozás: beállítások néven. A kék gomb helyett egy piros gomb fog látszódni kijelentkezés felirattal. Amennyiben valamelyik oldalon járunk a három hivatkozás közül, akkor annak a felirata egy fokkal sötétebb árnyalatot fog felvenni.

### Seasons

Ha a weboldalra térünk, akkor az első oldal, amivel szembe találjuk magunkat, az a szezonok listája. Ha vannak már szezonok tudunk köztük keresni, frissíteni, illetve bejelentkezett állapot esetén szűrni is. Ha nem jelenik meg adat, azt a weboldal egyértelműen jelezni fogja egy üzenettel. Minden egyes szezonnál meg fog jelenni a neve, a készítés dátuma, a létrehozó felhasználóneve, egy kedvencek és egy felfedezés gomb. Mobilnézet esetében azonban csak a név és a szív fog látszódni a reszponzivitás érdekében.

Bejelentkezés után 3 checkgomb fog megjelenni a jobb felső sarokban. Sorrendben a kedvencek, admin és moderátor gomb. Ha egyet benyomunk, akkor az szűrni fog, de ha kettőt, vagy mind a hármat, akkor hozzáadja azokat a szűrt elemeket, amik mondjuk a kedvencek és az admin uniója lesz. Természetesen szezonok duplikációja nem fordulhat elő. A kedvencek gombbal az általunk kedvelt szezonok fognak látszódni. Az admin gombbal azokat a szezonokat tekinthetjük meg, amelyeket mi hoztunk létre, vagy épp mi vagyunk a vezetői. A moderátor gombbal pedig azokat a szezonokat láthatjuk, amelyekhez hozzá vagyunk adva moderátorként.

// kép

Bejelentkezett állapotban lesz továbbá egy hozzáadás gomb, ahol egy ablak fog megnyílni egy űrlappal. Ez az egyetlen hely az egész programban, ahol új szezont adhatunk hozzá a rendszerhez. Meg kell adnunk a szezon nevét, illetve opcionálisan megadhatunk hozzá egy leírást is, hogy a felhasználók információt kapjanak a liga részleteiről. Amennyiben a nevet nem adtuk meg, vagy hibásan tettük mindezt, egy hibaüzenet figyelmeztetni fog minket erről. Ha sikeresen létrehoztuk a ligát, abban az esetben át leszünk irányítva annak az oldalára, hogy egyből láthassuk annak létezését.

// kép

Ami minden esetben meg fog jelenni bejelentkezés nélkül is, az a keresés, illetve a frissítés. Az előbbivel azonnali időben lehet szűrni a billentyű lenyomásával, míg az utóbbival frissíteni lehet az oldalt, ha esetleg valaki feltöltött az adatbázisba egy újabb szezont. Ha egy szezonra rányomunk át leszünk irányítva annak részletező oldalára.

## Season

Ez a program egyik legrészletesebb oldala. Rengeteg funkció található itt, több oldalon keresztül lehet mesélni arról, hogy milyen folyamatok zajlanak le itt egy-egy interakció során.