



Project Hephaistos

Készítette

Biliczki Bence

Fodor Valentin Sándor

Szegedi Dávid

Szoftverfejlesztő és -tesztelő szak

Témavezető

Kerényi Róbert Nándor

oktató

MISKOLC, 2024

Tartalomjegyzék

Bevezetés	4
Használt Technológiák	4
Backend	4
Adatbázis	4
Frontend	5
Fejlesztői Eszközök	5
Projekt Felépítése	5
Órarend Generátor Funkció	6
TimetableGenerator Osztály	6
TimetableGeneratorController Végpont	7
Felhasználói Élmény	8
1. Relációs adatkezelő rendszerek	9
1.1. Relációs adatbázis	9
1.1.1. A relációs adatbáziskezelő rendszerek	9
1.1.2. MySQL és MariaDB, valamint PHPMyAdmin	10
2. React Fejlesztői Környezet	11
2.1. Bevezetés	11
2.2. React Fejlesztői Környezet	11
2.2.1. React Telepítése és Projektindítás	11
2.2.2. Fejlesztői Eszközök	12
2.3. React Alapjai	12
2.3.1. Komponensek és JSX	12
Microsoft Azure Elérhetőség	12
Elérhetőség	13
2.3.2. Állapotkezelés a React-ben	14
2.3.3. Props	15
2.3.4. Komponensek Típusai	15
2.4. Jelszó Módosítása Funkció	16
2.4.1. ChangePasswordForm Komponens	16
2.4.2. Backend Integráció	17

2.4.3. Felhasználói Élmény	17
--------------------------------------	----

Bevezetés

A Project Hephaistos egy innovatív, egyetemisták számára készült órarend-tervező alkalmazás, amely célja, hogy hatékonyan támogassa a hallgatókat az órák, tanórán kívüli tevékenységek és egyéb kötelezettségek megszervezésében. Az alkalmazás C nyelven készült, és egy gépészmérnökhallgató igényei alapján lett fejlesztve, különös figyelmet szentelve a funkcionalitásnak és a felhasználóbarát felületnek. Az intelligens ütemezési megoldások lehetővé teszik az egyéni preferenciák, az egyetemi órarend és az időbeli ütközések figyelembevételét.

0.1. Használt Technológiák

A Hephaistos projekt különböző technológiákat használ a backend és a frontend fejlesztéséhez, valamint az adatbázis kezeléséhez.

0.1.1. Backend

A backend a .NET Core keretrendszert használja, amely egy nyílt forráskódú, platformfüggetlen eszközkészlet a modern, felhőalapú, internetkapcsolattal rendelkező alkalmazások fejlesztéséhez. A .NET Core segítségével a fejlesztők nagy teljesítményű és skálázható alkalmazásokat hozhatnak létre.

ASP.NET Core

Az ASP.NET Core a .NET Core keretrendszer része, amelyet webalkalmazások és API-k fejlesztésére használnak. Az ASP.NET Core előnyei közé tartozik a magas teljesítmény, a platformfüggetlenség és a moduláris felépítés. Az ASP.NET Core támogatja a modern webfejlesztési szabványokat és eszközöket, mint például a dependency injection, a middleware-ek és a RESTful API-k.

Az ASP.NET Core alkalmazások könnyen telepíthetők és skálázhatók, így ideálisak a felhőalapú környezetekben történő futtatásra. Az ASP.NET Core lehetővé teszi a fejlesztők számára, hogy gyorsan és hatékonyan hozzanak létre biztonságos és megbízható webalkalmazásokat.

0.1.2. Adatbázis

Az adatok tárolására MySQL adatbázist használunk. A MySQL egy népszerű, nyílt forráskódú relációs adatbázis-kezelő rendszer, amely nagy teljesítményt és megbízhatóságot kínál. A PHPMysqlAdmin eszközt használjuk az adatbázis kezelésére, amely egy webalapú felületet biztosít az adatbázisok kezeléséhez.

0.1.3. Frontend

A frontend fejlesztéséhez a React könyvtárat használjuk. A React egy népszerű JavaScript-könyvtár, amelyet a felhasználói felületek egyszerűbb fejlesztésére használnak. A React lehetővé teszi a fejlesztők számára, hogy dinamikus és interaktív felhasználói felületeket hozzanak létre.

React

A React egy komponens-alapú JavaScript könyvtár, amelyet a Meta fejlesztett ki és 2013-ban adtak ki. A React fő célja a felhasználói felületek egyszerűbb és hatékonyabb fejlesztése. A React lehetővé teszi a fejlesztők számára, hogy újrafelhasználható komponenseket hozzanak létre, amelyek könnyen karbantarthatók és bővíthetők.

A React egyik legfontosabb jellemzője a virtuális DOM (Document Object Model), amely javítja az alkalmazás teljesítményét azáltal, hogy minimalizálja a valódi DOM manipulációkat. A React támogatja a JSX (JavaScript XML) szintaxist, amely lehetővé teszi a HTML-szerű kód írását a JavaScript-ben, növelve a kód olvashatóságát és karbantarthatóságát.

A React ökoszisztémája számos kiegészítő könyvtárat és eszközt tartalmaz, mint például a React Router a kliensoldali útvonalkezeléshez és a Redux az állapotkezeléshez. Ezek az eszközök segítenek a fejlesztőknek hatékonyabb és skálázhatóbb alkalmazásokat létrehozni.

0.1.4. Fejlesztői Eszközök

A fejlesztés során különböző eszközöket használunk a hatékonyság növelése és a hibakeresés megkönnyítése érdekében:

- **Visual Studio Code (VSCode):** Egy népszerű kódszerkesztő, amely számos bővítménnyel rendelkezik, például az **ESLint** és **Prettier** segítségével.
- **React Developer Tools:** Egy böngészőbővítmény, amely lehetővé teszi a React komponensek és állapotok ellenőrzését a böngészőben.
- **PHPMyAdmin:** Egy webalapú eszköz a MySQL adatbázisok kezelésére.

0.2. Projekt Felépítése

A Hephaistos projekt három fő komponensből áll: backend, frontend és adatbázis. Az alábbiakban bemutatjuk ezeknek a komponenseknek a felépítését és funkcióit.

0.3. Órarend Generátor Funkció

A Project Hephaistos backend tartalmaz egy órarend generátor funkciót, amely segíti a felhasználókat az optimális órarend összeállításában. Ez a funkció figyelembe veszi a felhasználó által teljesített tantárgyakat, a tantárgyak előfeltételeit, valamint az időbeli ütközéseket.

0.3.1. TimetableGenerator Osztály

A `TimetableGenerator` osztály felelős az órarend generálásáért. Az osztály a következő fő metódust tartalmazza:

- **GenerateClosestTimetable**: Ez a metódus a megadott kreditszám alapján generál egy optimális órarendet. A metódus figyelembe veszi az elérhető tantárgyakat és az ütemezési konfliktusokat.

Az alábbi kód egy részletet mutat be a `GenerateClosestTimetable` metódusból:

```
1 public (IEnumerable<SubjectSchedule> Timetable, IEnumerable<
   Subject> OmittedSubjects) GenerateClosestTimetable(
2     int creditValue,
3     List<Subject> _availableSubjects,
4     List<SubjectSchedule> _existingSchedules
5 ) {
6     var selectedSubjects = new List<Subject>();
7     var omittedSubjects = new List<Subject>();
8     var selectedSchedules = new List<SubjectSchedule>();
9     int totalCredits = 0;
10
11     foreach (var subject in _availableSubjects) {
12         if (totalCredits + (subject.CreditValue ?? 0) >
13             creditValue) {
14             omittedSubjects.Add(subject);
15             continue;
16         }
17
18         var subjectSchedules = _existingSchedules.Where(s => s.
19             SubjectId == subject.Id).ToList();
20
21         if (!subjectSchedules.Any(schedule => IsOverlapping(
22             schedule, selectedSchedules))) {
23             selectedSubjects.Add(subject);
24             selectedSchedules.AddRange(subjectSchedules);
25         }
26     }
27 }
```

```

22         totalCredits += subject.CreditValue ?? 0;
23     } else {
24         omittedSubjects.Add(subject);
25     }
26 }
27
28 return (selectedSchedules, omittedSubjects);
29 }

```

0.3.2. TimetableGeneratorController Végpont

A `TimetableGeneratorController` egy API végpontot biztosít az órarend generálásához. A végpont a következőképpen működik:

- **POST** `/api/timetablegenerator/generate`: Ez a végpont fogadja a felhasználó által megadott kreditszámot, és visszaadja az optimális órarendet, valamint azokat a tantárgyakat, amelyek kimaradtak az ütemezésből.

Az alábbi kód egy részletet mutat be a végpont implementációjából:

```

1 [HttpPost("generate")]
2 [Authorize]
3 public async Task<IActionResult> GenerateTimetable([FromBody]
   int creditValue, [FromHeader(Name = "Authorization")] string
   Authorization) {
4     var userId = _jwtHelper.ExtractUserIdFromToken(
       Authorization);
5     var availableSubjects = GetAvailableSubjects(userId.Value);
6     var existingSchedules = _context.Subjectschedules
7         .Where(schedule => availableSubjects.Any(subject =>
           subject.Id == schedule.SubjectId))
8         .ToList();
9
10    var (timetable, omittedSubjects) = _timetableGenerator.
       GenerateClosestTimetable(creditValue, availableSubjects,
       existingSchedules);
11
12    return Ok(new {
13        Timetable = timetable.Select(schedule => new {
14            SubjectName = schedule.Subject?.Name,
15            DayOfWeek = schedule.DayOfWeek,
16            StartTime = schedule.StartTime,

```

```

17         EndTime = schedule.EndTime
18     }),
19     OmittedSubjects = omittedSubjects.Select(subject => new
20         {
21             SubjectName = subject.Name
22         })
23 }

```

0.3.3. Felhasználói Élmény

A funkció biztosítja, hogy a felhasználók egy könnyen értelmezhető órarendet kapjanak, amely tartalmazza az órák nevét, időpontját és napját. Az ütemezésből kimaradt tantárgyak külön listában jelennek meg, hogy a felhasználók tisztában legyenek azokkal a tantárgyakkal, amelyeket nem sikerült beilleszteni az órarendbe.

1. fejezet

Relációs adatkezelő rendszerek

1.1. Relációs adatbázis

Relációs adatbázisnak nevezzük a relációs adatmodell elvén létrehozott adatok összességét, a relációs adatmodell fogalomrendszerében meghatározott ún. relációk egy véges halmazát. Relációs adatbázisokat relációsadatbázis-kezelőkkel hozhatunk létre, szerkeszthetünk és törölhetünk.

1.1.1. A relációs adatbáziskezelő rendszerek

A relációs adatbáziskezelő rendszerek (RDBMS, azaz Relational Database Management System) olyan szoftverek, amelyek az adatokat táblák formájában kezelik. Minden tábla oszlopokból és sorokból áll, ahol az oszlopok különböző attribútumokat jelölnek, míg a sorok egyedi rekordokat reprezentálnak. Az RDBMS-ekben a táblák között relációk, azaz kapcsolatok hozhatók létre, így biztosítva az adatok egymáshoz kapcsolódó kezelését.

Az RDBMS-ek egyik legfontosabb jellemzője a strukturált lekérdezőnyelv, azaz az SQL (Structured Query Language), amely lehetővé teszi az adatok lekérdezését, módosítását és kezelését. Az adatokat kulcsok segítségével kapcsoljuk össze, mint például az elsődleges kulcs (primary key) és az idegen kulcs (foreign key). Az adatok normalizálásával csökkenthetők az ismétlődések, és fenntartható az adatintegritás. A relációs adatbáziskezelő rendszerek széles körben elterjedtek különböző üzleti és technológiai területeken, mivel lehetővé teszik az adatok hatékony tárolását és kezelését, valamint a nagy mennyiségű adat gyors lekérdezését.

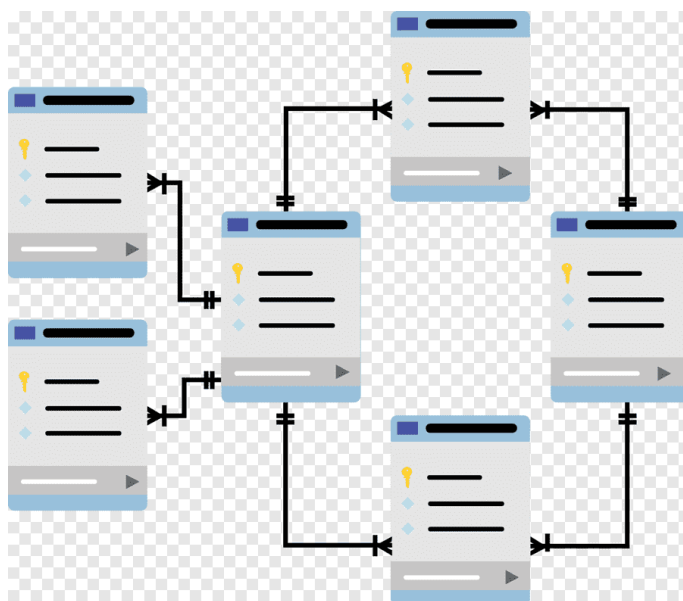
1.1.2. MySQL és MariaDB, valamint PHPMyAdmin

A MySQL és a MariaDB két népszerű, nyílt forráskódú relációs adatbáziskezelő rendszer. Mindkettő hasonló struktúrát és működési elveket követ, mivel a MariaDB a MySQL-ből származik. A MySQL-t eredetileg a Sun Microsystems fejlesztette ki, majd a Sun felvásárlása után az Oracle tulajdonába került. A MariaDB-t a MySQL eredeti fejlesztői hozták létre az Oracle irányításától való függetlenség megőrzése érdekében, és azóta külön fejlesztési irányt követ.

Mindkét rendszer könnyen kezelhető, nagy teljesítményt és skálázhatóságot kínál, így gyakran használják webalkalmazások és más dinamikus rendszerek adatbázisaként. A MySQL és a MariaDB támogatják a különféle táblastruktúrákat (például InnoDB és MyISAM), a tranzakciókezelést, valamint a különféle biztonsági mechanizmusokat, mint például a felhasználói jogosultságok finomhangolása.

A PHPMyAdmin egy népszerű, nyílt forráskódú webes eszköz, amelyet kifejezetten MySQL és MariaDB adatbázisok kezelésére fejlesztettek ki. Lehetővé teszi a felhasználók számára, hogy grafikus felületen keresztül, SQL parancsok írása nélkül menedzseljék az adatbázisokat. Ezzel az eszközzel könnyen létrehozhatók és módosíthatók táblák, kezelhetők a felhasználói jogosultságok, és futtathatók különböző SQL lekérdezések.

A PHPMyAdmin előnye, hogy böngésző alapú, így bármilyen eszközről könnyen elérhető, nem szükséges hozzá parancssori ismeret. A felhasználók könnyen átláthatják az adatbázis struktúráját, létrehozhatnak exportálási és importálási műveleteket, valamint automatikusan generált diagramok segítségével áttekinthetik a táblák közötti kapcsolatokat. Ezen kívül a PHPMyAdmin lehetőséget biztosít biztonsági mentések készítésére, így a felhasználók megőrizhetik és helyreállíthatják az adatokat.



1.1. ábra. Relációs adatbázis

2. fejezet

React Fejlesztői Környezet

2.1. Bevezetés

A React egy népszerű JavaScript-könyvtár, amelyet a felhasználói felületek egyszerűbb fejlesztésére használnak. Ezt a Meta fejlesztette, és 2013-ban adták ki. Azóta széles körben alkalmazzák a webes fejlesztésben, különösen az egyoldalas alkalmazások (SPA) készítéséhez. Ez a szakdolgozat a React fejlesztői környezetek felépítését, a szükséges eszközöket és a hatékony fejlesztéshez szükséges alapelveket vizsgálja.

2.2. React Fejlesztői Környezet

Ebben a fejezetben bemutatjuk a React fejlesztői környezet telepítését, a szükséges eszközöket, valamint a React alapjait, hogy a fejlesztés hatékony és eredményes legyen.

2.2.1. React Telepítése és Projektindítás

A React telepítéséhez először szükségünk van a Node.js és npm (Node Package Manager) telepítésére, amelyek segítségével kezelhetjük a JavaScript könyvtárakat és függőségeket. Miután a Node.js és az npm telepítve van, létrehozhatunk egy új React projektet az `create-react-app` eszközzel.

Az alábbi parancsokkal létrehozható egy új React projekt:

```
1 npx create-react-app my-app
2 cd my-app
3 npm start
```

A `create-react-app` gyorsan felállít egy alap React projektstruktúrát, amely a következő fájlokat tartalmazza:

- `src/` mappa: tartalmazza a forráskódot, beleértve a komponenseket és az alkalmazás logikáját.
- `public/` mappa: statikus fájlokat, például képeket és `index.html`-t tartalmazza.
- `node_modules/`: az összes szükséges függőséget tartalmazza.

2.2.2. Fejlesztői Eszközök

A React fejlesztéséhez számos fejlesztői eszköz áll rendelkezésünkre, amelyek növelhetik a termelékenységet és a hibakeresés hatékonyságát.

React Developer Tools

A React Developer Tools egy böngészőbővítmény, amely lehetővé teszi a React komponensek és állapotok ellenőrzését a böngészőben. Ez lehetővé teszi a hibakeresést, a komponensfa elemzését és az állapotok változásainak nyomon követését.

2.3. React Alapjai

A React egy összetevő-alapú megközelítést alkalmaz, amelyben az alkalmazás kisebb, újrafelhasználható komponensekből épül fel. Ez a moduláris felépítés segít a kód karbantartásában és a fejlesztés gyorsításában.

2.3.1. Komponensek és JSX

A React komponensek JavaScript függvények vagy osztályok, amelyek HTML-szerű kódot, úgynevezett JSX-et használnak. A JSX lehetővé teszi, hogy a JavaScript kódban HTML elemeket írjunk, ami növeli a kód olvashatóságát.

Microsoft Azure Elérhetőség

A Project Hephaistos alkalmazásunkat a Microsoft Azure felhőszolgáltatás segítségével publikáltuk, amely lehetővé teszi a skálázható és megbízható működést. Az Azure biztosítja az alkalmazás számára a szükséges infrastruktúrát, beleértve a webalkalmazás hosztolását, az adatbázis-kezelést és a biztonságos hozzáférést.

Az alkalmazás elérhető az alábbi linken: <https://hephaistos-backend-c6c5ewhraedvgzex.germanywestcentral-01.azurewebsites.net/index.html>

Azure Szolgáltatások Használata

A projekt során az alábbi Azure szolgáltatásokat használtuk:

- **Azure App Service:** Az alkalmazás hosztolására és futtatására használtuk. Ez egy teljesen menedzselt platform, amely lehetővé teszi a gyors telepítést és a folyamatos integrációt.
- **Azure SQL Database:** Az alkalmazás adatainak tárolására és kezelésére szolgál. Ez egy skálázható, biztonságos és megbízható relációs adatbázis-kezelő rendszer.
- **Azure Monitor:** Az alkalmazás teljesítményének és állapotának nyomon követésére használtuk. Ez segít az esetleges problémák gyors azonosításában és megoldásában.
- **Azure Active Directory (AAD):** A felhasználók hitelesítésére és az alkalmazás biztonságának növelésére használtuk.

Előnyök

A Microsoft Azure használata számos előnyt biztosít a projekt számára:

- **Skálázhatóság:** Az alkalmazás könnyen skálázható a felhasználói igények növekedésével.
- **Megerősített Biztonság:** Az Azure beépített biztonsági funkciói, például a titkosítás és a hozzáférés-szabályozás, garantálják az adatok védelmét.
- **Globális Elérhetőség:** Az Azure globális adatközpont-hálózata lehetővé teszi az alkalmazás gyors és megbízható elérését a világ bármely pontjáról.
- **Költséghatékonyság:** Az Azure rugalmas árazási modellje lehetővé teszi, hogy csak a ténylegesen használt erőforrásokért fizessünk.

A Microsoft Azure segítségével biztosítjuk, hogy a Project Hephaistos alkalmazás megbízhatóan és hatékonyan szolgálja ki a felhasználók igényeit.

```

1 import React from 'react';
2
3
4 function Welcome(props) {
5     return <h1>Hello, {props.name}</h1>;
6 }
7
8 export default Welcome;

```

2.3.2. Állapotkezelés a React-ben

A React komponensek állapotkezelése kulcsfontosságú a dinamikus alkalmazások létrehozásában. Az állapot segítségével megőrizhetjük a komponens adatainak aktuális állapotát.

useState Hook

A `useState` egy hook, amely lehetővé teszi az állapotkezelést a funkcionális komponensekben. Az alábbi példa egy egyszerű számlálót mutat be, amely a `useState` használatával kezeli az állapotot:

```

1 import React, { useState } from 'react';
2
3 function Counter() {
4     const [count, setCount] = useState(0);
5
6     return (
7         <div>
8             <p>You clicked {count} times</p>
9             <button onClick={() => setCount(count + 1)}>
10                 Click me
11             </button>
12         </div>
13     );
14 }
15
16 export default Counter;

```

2.3.3. Props

A React komponensek közötti kommunikációt az úgynevezett **props** (properties) segítségével valósítják meg. A **props** adatokat továbbítanak az egyik komponensből a másikba.

```
1 function Greeting(props) {  
2     return <h1>Hello, {props.name}</h1>;  
3 }  
4  
5 function App() {  
6     return <Greeting name="React Developer" />;  
7 }  
8  
9 export default App;
```

2.3.4. Komponensek Típusai

A React komponensek két fő típusa:

- **Funkcionális komponensek**: egyszerű JavaScript függvények, amelyek a **props**-ot paraméterként veszik át és JSX-et adnak vissza.
- **Osztály alapú komponensek**: osztályok, amelyek bonyolultabb állapotkezelést és életciklus-módszereket tesznek lehetővé.

2.4. Jelszó Módosítása Funkció

A React frontend alkalmazás tartalmaz egy jelszó módosítási funkciót, amely lehetővé teszi a felhasználók számára, hogy biztonságosan megváltoztassák a jelszavukat. Ez a funkció a következő komponensekből áll:

2.4.1. ChangePasswordForm Komponens

A ChangePasswordForm egy React komponens, amely biztosítja a jelszó módosításához szükséges űrlapot. Az űrlap tartalmaz mezőket a régi jelszó, az új jelszó és az új jelszó megerősítésének megadásához. Az alábbi kód egy részletet mutat be a komponensből:

```
1      import React, { useState } from 'react';
2
3      function ChangePasswordForm() {
4          const [oldPassword, setOldPassword] = useState(
5              '');
6          const [newPassword, setNewPassword] = useState(
7              '');
8          const [confirmPassword, setConfirmPassword] =
9              useState('');
10
11          const handleChangePassword = () => {
12              if (newPassword !== confirmPassword) {
13                  alert('Az új jelszavak nem
14                      egyeznek!');
15                  return;
16              }
17              // Jelszó módosítási logika
18          };
19
20          return (
21              <form>
22                  <input type="password" placeholder="Régi jelszó"
23                      " onChange={(e) => setOldPassword(e.target.
24                          value)} />
25                  <input type="password" placeholder="Új jelszó"
26                      onChange={(e) => setNewPassword(e.target.
27                          value)} />
28                  <input type="password" placeholder="Új jelszó
29                      megerősítése" onChange={(e) =>
30                          setConfirmPassword(e.target.value)} />
```



```

21         <button type="button" onClick={
                handleChangePassword}>Jelszó módosítása</
                button>
22     </form>
23     );
24 }

```

2.4.2. Backend Integráció

A jelszó módosítási funkció a backend API-val kommunikál, hogy ellenőrizze a régi jelszót és frissítse az új jelszót az adatbázisban. Az API végpontja a következő:

```

1 POST https://localhost:5001/change-password
2 Headers: {
3     Authorization: Bearer <token>
4 }
5 Body: {
6     "oldPassword": "current_password",
7     "newPassword": "new_password"
8 }

```

2.4.3. Felhasználói Élmény

A funkció biztosítja, hogy a felhasználók értesítést kapjanak a sikeres vagy sikertelen jelszó módosításról. A hibák, például a nem egyező jelszavak vagy a helytelen régi jelszó, azonnal megjelennek az űrlapon.