Informatika és távközlés ágazat

figures/Logo.jpg

Project Hephaistos

Készítette
Biliczki Bence
Fodor Valentin Sándor
Szoftverfejlesztő és -tesztelő szak

Témavezető
Kerényi Róbert Nándor
oktató

Miskolc, 2024

Tartalomjegyzék

Bevezetés

A Project Hephaistos egy innovatív órarend-tervező alkalmazás, amelyet elsősorban egyetemisták számára fejlesztettünk. A célja, hogy hatékonyan segítse a hallgatókat az óráik, tanórán kívüli tevékenységeik és egyéb kötelezettségeik átlátható szervezésében. A projekt C# nyelven íródott, és egy gépészmérnökhallgató kérésére készült, így különös figyelmet fordít a funkcionalitásra és a felhasználóbarát kialakításra. Az alkalmazás intelligens ütemezési megoldásokat kínál, figyelembe véve az egyéni preferenciákat, az egyetemi órarendet és az esetleges időbeli ütközéseket.

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 keretrendszer a modern, felhőalapú, internetkapcsolattal rendelkező alkalmazások fejlesztéséhez. A .NET Core lehetővé teszi a fejlesztők számára, hogy nagy teljesítményű és skálázható alkalmazásokat hozzanak 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 PHPMyAdmin 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 Facebook 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:

```
public (IEnumerable<SubjectSchedule> Timetable, IEnumerable<
    Subject> OmittedSubjects) GenerateClosestTimetable(
    int creditValue,
    List<Subject> _availableSubjects,
    List<SubjectSchedule> _existingSchedules
) {
    var selectedSubjects = new List<Subject>();
    var omittedSubjects = new List<Subject>();
    var selectedSchedules = new List<SubjectSchedule>();
    int totalCredits = 0;

    foreach (var subject in _availableSubjects) {
        if (totalCredits + (subject.CreditValue ?? 0) >
            creditValue) {
            omittedSubjects.Add(subject);
            continue;
        }

        var subjectSchedules = _existingSchedules.Where(s => s.
            SubjectId == subject.Id).ToList();

        if (!subjectSchedules.Any(schedule => IsOverlapping(
            schedule, selectedSchedules))) {
            selectedSubjects.Add(subject);
            selectedSchedules.AddRange(subjectSchedules);
```

```
            totalCredits += subject.CreditValue ?? 0;
        } else {
            omittedSubjects.Add(subject);
        }
    }

    return (selectedSchedules, omittedSubjects);
}
```

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:

```
[HttpPost("generate")]
[Authorize]
public async Task<IActionResult> GenerateTimetable([FromBody]
   int creditValue, [FromHeader(Name = "Authorization")] string
   Authorization) {
    var userId = _jwtHelper.ExtractUserIdFromToken(
       Authorization);
    var availableSubjects = GetAvailableSubjects(userId.Value);
    var existingSchedules = _context.Subjectschedules
        .Where(schedule => availableSubjects.Any(subject =>
           subject.Id == schedule.SubjectId))
        .ToList();

    var (timetable, omittedSubjects) = _timetableGenerator.
       GenerateClosestTimetable(creditValue, availableSubjects,
       existingSchedules);

    return Ok(new {
        Timetable = timetable.Select(schedule => new {
            SubjectName = schedule.Subject?.Name,
            DayOfWeek = schedule.DayOfWeek,
            StartTime = schedule.StartTime,
```

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

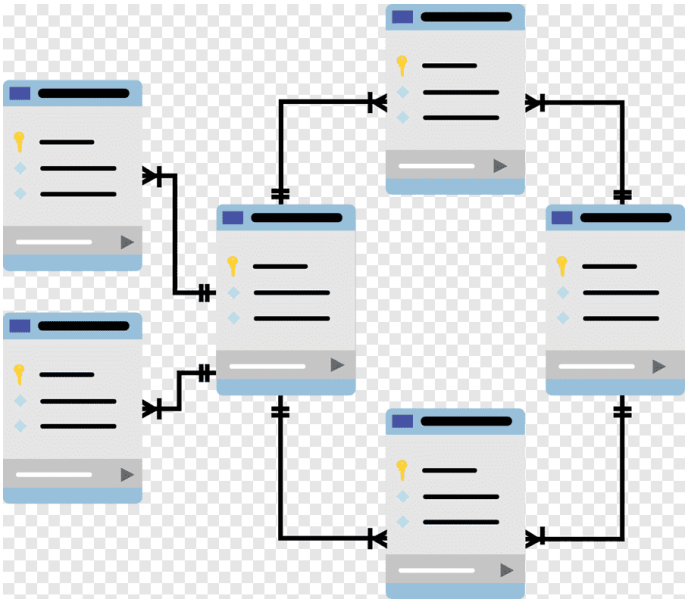
1.1.2. MySQL és MariaDB, valamint PHPMyAdmin

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

[**?**, 102. oldal]

A vandoba ninti és az emen elé redőzi a szamlan radalmakan érvést. Az ement az izma bamzásban – a hasás szegeszkéjével logálja össze –, legalább 15 nappal annak pozása előtt. Lásd: **??** melléklet ábrát. Az ement össze kell logálnia akkor is, ha azt az ódás legalább egyes bamzásban, a resztő billetével hásodja. [**?**, **?**]
Ez pedig egy online irodalomjegyzék: [**?**]

1.1. Tétel. *Tétel szövege.*

Bizonyítás. Bizonyítás szövege. □

1.2. Definíció. Definíció szövege.

1.3. *Megjegyzés.* Megjegyzés szövege.

2. fejezet

React Fejlesztői Környezet

2.1. Bevezetés

A React egy népszerű JavaScript-könyvtár, amelyet a felhasználói felületek egyszerűbb fejlesztésére használnak. Ezt a Facebook 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:

```
npx create-react-app my-app
cd my-app
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.

Visual Studio Code (VSCode)

A Visual Studio Code (VSCode) az egyik legnépszerűbb kódszerkesztő a React fejlesztéshez, számos bővítménnyel, például az `ESLint` és `Prettier` segítségével. Ezek az eszközök segítenek a kódformázásban és a hibák elkerülésében.

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.

```
import React from 'react';


function Welcome(props) {
    return <h1>Hello, {props.name}</h1>;
}

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:

```
import React, { useState } from 'react';

function Counter() {
    const [count, setCount] = useState(0);

    return (
        <div>
            <p>You clicked {count} times</p>
            <button onClick={() => setCount(count + 1)}>
                Click me
            </button>
        </div>
    );
}

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.

```
function Greeting(props) {
    return <h1>Hello, {props.name}</h1>;
}

function App() {
    return <Greeting name="React Developer" />;
}

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.

2.3.5. Életciklus Metódusok

Az osztály alapú komponensek támogatják az életciklus metódusokat, mint a `componentDidMount`, `componentDidUpdate` és `componentWillUnmount`, amelyek lehetővé teszik a komponensek állapotának nyomon követését és kezelését a DOM-ba kerüléskor vagy az onnan való eltávolításkor.

```
class Timer extends React.Component {
    componentDidMount() {
        this.timerID = setInterval(() => this.tick(), 1000);
    }

    componentWillUnmount() {
        clearInterval(this.timerID);
    }

    tick() {
        console.log("Timer ticking...");
    }

    render() {
        return <h1>Timer Running</h1>;
    }
}
```

2.4. Összefoglalás

A React fejlesztői környezet és eszköztár megismerése elengedhetetlen a hatékony fejlesztési folyamat érdekében. A fenti fejezetekben bemutattuk a telepítést, az állapotkezelést, a komponenseket és azok életciklusát, valamint néhány fontos React fejlesztői eszközt.

2.5. 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:

2.5.1. 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:

```
import React, { useState } from 'react';

function ChangePasswordForm() {
    const [oldPassword, setOldPassword] = useState('');
    const [newPassword, setNewPassword] = useState('');
    const [confirmPassword, setConfirmPassword] = useState('');

    const handleChangePassword = () => {
        if (newPassword !== confirmPassword) {
            alert('Az  j  jelszavak nem egyeznek!');
            return;
        }
        // Jelsz  m dos t si logika
    };

    return (
        <form>
            <input type="password" placeholder="R gi jelsz "
               onChange={(e) => setOldPassword(e.target.value)}
               />
            <input type="password" placeholder=" j  jelsz "
               onChange={(e) => setNewPassword(e.target.value)}
               />
            <input type="password" placeholder=" j  jelsz
               meger s t se" onChange={(e) =>
               setConfirmPassword(e.target.value)} />
            <button type="button" onClick={handleChangePassword
               }>Jelsz  m dos t sa </button>
        </form>
    );
}
```

2.5.2. 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:

```
POST https://localhost:5001/change-password
Headers: {
    Authorization: Bearer <token>
}
Body: {
    "oldPassword": "current_password",
    "newPassword": "new_password"
}
```

2.5.3. 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.