**Eseményvezérelt programozás**

**Beadandó projectmunka**

**Csapatunk tagjai:**

Harkai Martin - H1RWPP – h1rwpp@inf.elte.hu

Rába Krisztián - YP5Y25 - yp5y25@inf.elte.hu

Vass-Horváth Balázs – B0XUPT - b0xupt@inf.elte.hu

**Választott téma:**

**IK WEBSHOP**

Tárgykód: IKSEK-17EVPROGEG

Gyakorlatvezető: Soós Sándor

# Tartalomjegyzék

# Fejlesztői információk

## Felhasznált nyelvek, technológiák, szoftverek

### Java – Spring keretrendszer

A java napjaink egyik legnépszerűbb objektumorientált programozási nyelve. Fejlesztése egészen az 1990-es évek elejére nyúlik vissza, amikoris még a Sun Microsystems fejlesztette. 2009-ben az Oracle cég tulajdonába került egy felvásárlás eredményeképpen, és napjainkban is ők felelnek a fejlesztéséért. Érdekesség, hogy egy kávéscsésze lett az ikonja a nyelvnek, melyet kávézás közben találtak ki. Népszerűségét valószínűleg annak köszönheti, hogy platformtól függetlenül fejleszthetünk alkalmazásokat a segítségével. A platformfüggetlenséget egyedi módon valósította meg a cég. Az általunk megírt forráskódból a fordító valójában nem egy futtatható (exe) állományt hoz létre, hanem úgynevezett java bájtkódot, mely a Java Virtual Machine (JVM) virtuális gépen fog futni. A megvalósítás előnye a hardware, és platform függetlenség, ugyanakkor hátránya, hogy ez lassítja a programunk futását, valamint növeli az erőforrásigényét. Annak érdekében, hogy java alkalmazást fejleszthessünk szükségünk van az úgynevezett Java Development Kit-re (JDK), mely a fejlesztői eszközök mellett tartalmazza a Java Runtime Environment-et (JRE) is.

A Spring egy nyílt forráskódú, java keretrendszer, melynek első korai verziója 2003-ban jelent meg, míg az első stabil kiadást 2004-ben adták ki. A java nyelvvel együtt a keretrendszer is folyamatos fejlődésnek örvend. Kezdetben csak néhány modul volt elérhető, és asztali alkalmazások fejlesztését gyorsította. Napjainkban már több-száz kiegészítő modult importálhatunk be, leggyakrabban webes alkalmazások fejlesztésére használjuk. Néhány hasznos modul például az adatbázis-kezelést gyorsító JDBC, vagy az autentikációhoz használgató oAuth2 modul, melyek használatával rengeteg időt nyerhetünk egy alkalmazás fejlesztésekor. Fontos jellemzője továbbá, hogy jól használható dokumentációja van, ezért a java nyelv ismeretében könnyen nem fog gondot okozni a használata. Spring alkalmazást ma már néhány kattintással is létrehozhatunk, csak a project típusát (maven / gradle), nyelvét (java / kotlin), a spring verzióját, és a project adatait (név, leírás, package név, java verzió) kell megadnunk, valamint kiválaszthatjuk a használni kívánt kiegészítő modulokat.

Java - <https://hu.wikipedia.org/wiki/Java_(programoz%C3%A1si_nyelv)>

Spring - <https://hu.wikipedia.org/wiki/Spring_keretrendszer>

### MySql 8.0

A MySql az Oracle Corporation által fejlesztett SQL alapú relációs adatbázis szerver. A MySql mára az egyik legelterjedtebb adatbázis-kezelő olyan okok miatt, mint például a nyílt forráskód, a megbízhatóság, a stabilitás, valamint az, hogy platform független, tehát elérhető többek között Windows, Linux, MacOS X, FreeBSD, és szinte minden további operációs rendszeren. Az adatbázis kezeléséhez szükségünk van továbbá egy adatbázis kezelő szoftverre, mint például a Navicat, vagy a MySql Workbench. MySql adatbázis kezelő szoftverek szinte minden operációs rendszerre léteznek, még Androidra, vagy IOS-re is. Alternatívájaként érdemes megemlíteni az MSSQL-t, amely hasonló hatékonysággal képes ellátni az adatbázis feladatokat, azonban mivel a MySql-t az Oracle cég fejleszti, ezért (is) érdemesebb java alkalmazásokhoz

MySql - https://hu.wikipedia.org/wiki/MySQL

### React JS

A React vagy más néven ReactJS egy ingyenes, open-source front-end Javascript könyvtár, aminek a segítségével könnyedén hozhatunk létre felhasználói felületeket(User Interface) komponensekkel. Előfeltétele, hogy rajta legyen a Node a helyi rendszeren, ha ez teljesül, akkor akár egy parancs kiadásával a cmd-ben vagy bash-ben létre is hozhatjuk a React projektünket. Nagy szeretettel használják az SPA(Single-Page Application) miatt is, hiszen a kliensnek nem kell újra töltenie az oldalt. A React state-eket vagyis „állapotokat” kezel és jeleníti meg a DOM-on(Document Object Model). A state-k automatikusan frissülnek, ha változás történik rajtuk, anélkül, hogy újra betöltenénk az oldalt. A komponensek újra felhasználhatóak(Pl: Ha több oldalon akarjuk ugyanazt a tartalmat megjeleníteni), de még úgymond props-ot is kaphat, ami azért jó, mert a komponensek között értékeket tudunk átadni. A komponenseket exportálni kell, hogy azt másik komponensben be tudjuk importálni. Léteznek függvény és osztály-alapú komponensek, amik abban különböznek, hogy a függvényes komponensekkel hook-okat, magyarosítva „horgonyokat” használhatunk, viszont az osztály alapúaknál már nem megengedett. Beépített Hook-ok: useState, useEffect, useContext, amiket legtöbbször a statek és a mellékhatások kezelésére használják a napokban. Talán ezért is kedveltebb a függvényes megoldás. Vannak Lifecycle metódusok is, amikkel meghatározhatjuk, hogy a komponensek frissüljenek-e. A react egyébként a JSX szintakszist használja, ami nagyban hasonlít a JavaScript-hez, egyik előnye, hogy elfogadja a HTML tageket, de még a CSS-t is. Az npm-el telepíthetünk további JS modulokat (a mi projektünk pl a react-router-dom-ot használja a route-olás megoldásához), amik a node\_modules mappában vannak tárolva. Fontos: A node\_modules mappát mindig ignoráljuk, amikor commit-olunk! Ha többen dolgozunk egy react projekten, akkor indítás előtt(npm start) érdemes beiírni az npm install-t, hogy lehetőleg elkerüljük az incidenseket.

### GitHub

A GitHub a Microsoft egy leányvállalataként a verziókezelés, verziókövetés lehetőségét biztosítja. A projectmunka elkészítése közben fontos volt, hogy párhuzamosan fejleszthessük az alkalmazásunkat. Ezen felül lehetőséget biztosít számunkra, hogy egy esetleges hibás fejlesztést követően visszaállítsuk a kódot egy régebbi verziójára, és nem utolsó sorban egy esetleges adatvesztés következtében is vissza tudom nyerni a korábbi munkánkat. Segítségével nyomon lehet követni, hogy miként fejlődik a program, mikor milyen fejlesztéseket, újítások kerülnek bele. Bár legfőképpen forráskódjainkat tároljuk itt, de lehetőség van például dokumentáció, Wikipédia, integrációs könyvtárak (LIB-ek), és még rengeteg virtuálisadat tárolására, nyomon követésére. A GitHub továbbá lehetőséget nyújt számunkra kisebb weboldalak tárolására, valamint kódjaink publikálására, és még megannyi hasznos funkcióra, melyeket szinte a végtelenségekig lehetne sorolni. Alternatívaként rengeteg lehetőséget felsorolhatnék, de talán a legismertebb az azonos alapokra épülő GitLab.

GitHub - <https://en.wikipedia.org/wiki/GitHub>

## Használt szoftverek, fejlesztői környezetek

### SpringToolSuite4

A SpringToolSuite4 egy Eclipse-re épülő spring (java) fejlesztői környezet, a VMware cégtől. A többi fejlesztői környezettel szemben hatalmas előnye, hogy segítségével letölthető modulokat építhetünk be a projectünkbe automatikusan, nem kell ezt manuálisan elvégeznünk. Teljes értékű java fejlesztői környezet is, mivel Eclips alapra épül, ezáltal tökéletesen használhatjuk az alkalmazásunk fejlesztése során.

### Navicat

A Navicat egy 2002-ben, a PremiumSoft CyberTech Ltd. által bemutatott grafikus adatbáziskezelő szoftver, mely támogatja többek között a MySql-t, MariaDB-t, MongoDB-t, Oracle, SQLite-ot, PostgreSQL-t és az MSSQL-t is. A Navicat elérhető Linux, Windows, és MacOS X rendszerekre is. Felhasználó barát felületének köszönhetően átlátható, egyszerűen 8 használható. A hagyományos lekérdezéseken túl adatbázis modellezésre, tervezésre, és adatbázis kapcsolatok kialakítására is egyaránt alkalmas. Az adatbáziskezeléshez alternatíva lehet az ingyenesen elérhető MySql Workbench.

Navicat - <https://en.wikipedia.org/wiki/Navicat>

### GitHub Desktop

A GitHub Desktop a GitHub saját fejlesztéstű szoftvere. Célja, hogy a GitHub-on kezelt munkáinkat könnyebben, átláthatóbban, a szóban forgó asztali alkalmazással elérhessük. Elérhető Windows, valamint MacOS, rendszerekre, viszont amennyiben Linux alapú rendszerünk van, sajnos más alternatívát kell keresnünk. Fontos megemlíteni, hogy a GitHub mobilos alkalmazást is fejlesztett azonos a célra, amely elérhető androidos, és IOS-es eszközökre is. A program lehetőséget biztosít a GitHub verziókövető rendszer legtöbb funkciójának elérésére, miután bejelentkeztünk a GitHub fiókunkkal. Ezen funkciók közül a leglényegesebbek például a projektünk klónozása az adott számítógépre, a módosítások feltöltése, valamint a korábbi verziók visszaállítása, de összehasonlíthatjuk vele a programunk két verzióját, valamint megnézhetjük, mikor mit módosítottunk rajta. Bár a Visual Studio rendelkezik beépített funkcióval, mely segítségével letölthetjük a projektünk friss verzióját, valamint feltölthetjük a módosításainkat, ez messze elmarad funkcionalitás, felhasználói felület, és kezelés terén is a felhasználóbarát GitHub Desktop-tól. Hasonló alkalmazásként megemlíthető például a Fork.

### Visual Studio Code

## Adatbázis, tárolt adatok, adatvédelem

A program működéséhez szükséges adatok tárolásáért egy MySql adatbázis felel. A tárolt adatok közötti kapcsolat az adatbázis ábrán látható. Adatintegritási problémák elkerülése végett 3 szinten (Kliens – Szerver - Adatbázis) is szigorú ellenőrzésen mennek keresztül az adatok az eltárolás előtt. A kliens oldali ellenőrzés célja főleg a felhasználó informálása, és a felesleges adatforgalom generálásának elkerülése a célja. A legfőbb ellenőrzés szerveroldalon, valamint az adatbázisban zajlik, ahol az egyes táblák közötti kapcsolatok kialakításán túl az adatbázisnak küldött adatok ellenőrzése (Pl.: karakterlánc hossza, numerikus típusok min/max értéke stb.) is megtörténik (CONSTRAINT).

Egy olyan alkalmazásnál, ahol a felhasználó személyesebb adatait is eltároljuk (Pl.: név, szállításhoz lakcím) kiemelkedően fontos az adatvédelem, hogy jogosulatlan személyek ne férhessenek hozzá az ügyfelek adataihoz. Ennek érdekében szükséges, hogy az alkalmazáson belül is helyén kezeljük a jogosultságokat (Például egy átlag felhasználó ne férhessen hozzá a többi felhasználó adataihoz), valamint megfelelően védjük az adatbázisunkat a külső támadásoktól. A külső behatolások megakadályozása érdekében első, és legfontosabb, hogy az adatbázisunk felhasználói megfelelően legyen beállítva (minden felhasználó csak a szükséges adatbázisokhoz/táblákhoz férjen hozzá), valamint erős jelszóval védjük a felhasználót. Tovább szűkíthetjük a kockázatot SSH tunnel használatával, valamint IP cím szűréssel. Webes alkalmazás révén, a kliens-szerver közötti kommunikáció biztonsága érdekében HTTPS protokoll használatát kell előnyben részesíteni a HTTP protokollal szemben.

A felhasználók jelszava minden esetben SHA256-os titkosítással kerül eltárolásra, mely napjaink egyik legbiztonságosabb megoldása.