FELADATKIÍRÁS

A feladatkiírást a tanszéki adminisztrációban lehet átvenni, és a leadott munkába eredeti, tanszéki pecséttel ellátott és a tanszékvezető által aláírt lapot kell belefűzni (ezen oldal *helyett*, ez az oldal csak útmutatás). Az elektronikusan feltöltött dolgozatban már nem kell beleszerkeszteni ezt a feladatkiírást.

Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Hálózati Rendszerek és Szolgáltatások Tanszék

Leltározási feladatokat segítő, grafikus felülettel rendelkező adatbázis tervezése és fejlesztése

Szakdolgozat

Készítette
Király Bálint Martin

Konzulens
Schulcz Róbert

2020. november 8.

Tartalomjegyzék

Kivonat **i**

Abstract **ii**

1. Bevezetés **1**

2. Feladat specifikáció **2**
2.1. Nem funkcionális követelmények 2
2.2. Funkciónális követelmények 2

3. Választott technologiák **3**
3.1. GraphQL 3
3.2. TypeScript 3
3.3. Frontend 4
3.4. Backend 4
3.5. Adatbázis 4

4. Architektúra **5**
4.1. Adatbázis séma 5
4.2. Backend felépítése 5
4.3. Frontend felépítése 5
4.4. A teljes kép 5

5. Wireframe-ek **6**

6. Tesztelés **7**
6.1. Fronted tesztelés 7

7. Összefoglalás **8**

Köszönetnyilvánítás **9**

Irodalomjegyzék **10**

Függelék **11**

HALLGATÓI NYILATKOZAT

Alulírott *Király Bálint Martin*, szigorló hallgató kijelentem, hogy ezt a szakdolgozatot meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző(k), cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy autentikált felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetővé.

Budapest, 2020. november 8.

Király Bálint Martin
hallgató

Kivonat

Jelen dokumentum egy diplomaterv sablon, amely formai keretet ad a BME Villamosmérnöki és Informatikai Karán végző hallgatók által elkészítendő szakdolgozatnak és diplomatervnek. A sablon használata opcionális. Ez a sablon LaTeX alapú, a *TeXLive* TeX-implementációval és a PDF-LaTeX fordítóval működőképes.

Abstract

This document is a LaTeX-based skeleton for BSc/MSc theses of students at the Electrical Engineering and Informatics Faculty, Budapest University of Technology and Economics. The usage of this skeleton is optional. It has been tested with the *TeXLive* TeX implementation, and it requires the PDF-LaTeX compiler.

1. fejezet

Bevezetés

A félév során a feladatom egy olyan grafikus rendszerrel ellátott leltár rendszer tervezése és fejlesztése volt, melynek segítségével az alapvető leltári funkciókon felül könnyedén behatárolhatjuk a leltárba vett eszközök pontos helyzetét a raktárunkon belül egy grafikus "térkép" segítségével.

A dolgozatom során első körben egy részletes specifikációt készítettem, melyben taglaltam az alkalmazással szemben támasztott funkcionális és nemfunkcionális követelményeket. Ezután megterveztem a webalkalmazás felhasználói felületét, egyszerű wireframe-ek segítségével.

A specifikáció és a wireframe-ek elkészítése után kiválasztottam a használni kívánt technológiákat és megterveztem az alkalmazás arhitekturális felépítését.

Az alkalmazás két fő részből áll, frontend és backend. Utóbbi tartalmazza az üzleti logikát és az adatbázis kommunikációt, míg a frontend az adatok lekéréséért és megjelenítéséért felel, valamint a felhasználói interakciók által eljuttatja a módosításokat a backend részére. Ezeket később részletesen taglalom.

Az alkalmazás tervezésén és fejlesztésén kívül az üzemeltetés előkészítését is elvégeztem.

Végül a tesztelésre fektettem a hangsúlyt. Fejlesztés közben igyekeztem figyelni, hogy minél jobb tesztlefedettséget érjek el, azonban a fejlesztés végeztével nem voltam elégedett az eredménnyel, így további teszteket és metrikákat készítettem.

2. fejezet

Feladat specifikáció

2.1. Nem funkcionális követelmények

Lorem ipsum

2.2. Funkciónális követelmények

Lorem ipsum dolor set

3. fejezet

Választott technologiák

Annek a fejezetnek a keretein belül a felhasznált technologiákat szeretném bemutatni. Elöször azokat a részeket mutatom be, melyek az alkalmazás több részét is lefedik, majd a backend, a frontend és az adatbázis rétekeget mutatom be részletesen.

3.1. GraphQL

A graphQL egy lekérdező nyelv, amely a jelenleg elterjedt REST API-s megoldásokat próbálja leváltani/kiegészíteni. A megszokott REST API-val ellentétben GraphQL-nél csak egyetlen egy végpont létezik, valamint csak POST típusú HTTP kéréseket használunk.

Az összes kérést erre a végpontra küldjük a megfelelő tartalommal, melyet a POST kérés törzsében (body) helyezünk el.

A bevett REST API-s megoldással szembeni hatalmas előnye, hogy mindig azt kapjuk amit kérünk. A POST kérés törzsében elhelyezett GraphQL operation pontosan meghatározza, hogy milyen entitások milyen tulajdonságait szeretnénk visszakapni. Ez a GraphQL operation nagyon hasonlít a JSON formátumra, azonban egy-két dologban eltér attól. Lehetőségünk van több entitásból is adatot lekérni egyetlen kéréssel, így csökkentve a HTTP üzenetek számát.

A kéréseket minden esetben egy (vagy több) úgynevezett resolver szolgálja ki nekünk.

A resolvereiből 3 fő típust különböztetünk meg Query, Mutation és Subscription.

Definíció 1 (Query). Adatok lekérésére szolgál ▪

Definíció 2 (Mutation). Ahogy a nevéből is következtethetünk rá főként adatok módosítására és létrehozására szolgál ▪

Definíció 3 (Subscription). A standard GraphQL implementáció tartalmazza a websocket kommunikációt is. A subscription-ök segítségével lehetősége van a kliensnek feliratkozni bizonyos eseményekre, melyek bekövetkeztéről azonnal értesül socket kapcsolaton keresztül. ▪

3.2. TypeScript

A TypeScript egy - a Microsoft által fejlesztett - nyílt forráskódú nyelv, amely JavaScript-et egészíti ki statikus típus definíciókkal. Mondhatjuk, hogy a JavaScript egy superset-je.

A típusok segítségével hamarabb észrevehetjük a hibákat az alkalmazásunkban. Azonban fontos megjegyezni, hogy a típusok definiálása opcionális, ezért TypeScript mellett érdemes valamilyen linter-t használni, amely figyelmezteti a programozót ha elmulasztja a típusdefiníciók használatát.

Minden valid JavaScript kód egy valid TypeScript kód is, ez részben az elhagyható típusdefiníciók miatt igaz.

Annak érdekében, hogy probléma nélkül futtathassuk a TypeScript kódunkat a böngészőkben minden kódot JavaScript-re transzformálunk. Erre több megoldás is létezik, ilyen például a Babel vagy a TypeScript complier.

A NodeJS-nek köszönhetően használhatjuk backend oldali nyelvként is, így a frontend és a backend közös nyelvet használhat, amely akár a kódmegosztás lehetőségét is felveti.

3.3. Frontend

Lorem ipsum

3.4. Backend

Lorem ipsum

3.5. Adatbázis

Lorem ipsum

4. fejezet

Architektúra

Lorem ipsum

4.1. Adatbázis séma

Lorem ipsum

4.2. Backend felépítése

Lorem ipsum

4.3. Frontend felépítése

Lorem ipsum

4.4. A teljes kép

Lorem ipsum

5. fejezet

Wireframe-ek

Lorem ipsum

6. fejezet

Tesztelés

Lorem ipsum

6.1. Fronted tesztelés

Lorem ipsum

7. fejezet

Összefoglalás

Lorem ipsum

Köszönetnyilvánítás

Ez nem kötelező, akár törölhető is. Ha a szerző szükségét érzi, itt lehet köszönetet nyilvánítani azoknak, akik hozzájárultak munkájukkal ahhoz, hogy a hallgató a szakdolgozatban vagy diplomamunkában leírt feladatokat sikeresen elvégezze. A konzulensnek való köszönetnyilvánítás sem kötelező, a konzulensnek hivatalosan is dolga, hogy a hallgatót konzultálja.

Irodalomjegyzék

Függelék