Eötvös Loránd Tudományegyetem

Informatikai Kar

Programozási nyelvek és Fordítóprogramok Tanszék

Kisvállalatokat segítő webáruház Angular keretrendszerben

Témavezető:
Fekete Anett
PhD hallgató, MSc

Szerző:
Magyar Dorina
programtervező informatikus BSc

Budapest, 2021

Az eredeti szakdolgozati / diplomamunka témabejelentő helye.

Tartalomjegyzék

1. fejezet

Bevezetés

A **WebBeauty**(továbbiakban **WB**) lehetőséget kínál a kisvállalkozók által gyártott termékek bemutatására és árusítására. Mivel napjainkban menőt a kereslet a személyes webáruházak létrehozása iránt, ahol a saját termékeiket kívánják értékesíteni, ezt pedig a **WB** megfelelően kiszolgálja. Az webalkalmazás nem csak a felhasználók számára könnyen kezelhető, hanem egyben a tulajdonosnak is. Lehetőségük van arra, hogy egyszerűen menedzselhessék termékeiket és kapcsolatot tartsanak a lehetséges vásárlókkal.

Számomra a témaválasztás célja egy személyesen ismert vállalkozó megkeresésén alapult, aki szívesen értékesítene az általam készített webáruházon keresztül a termékeit. Az alap problémát az vetette fel részemről, hogy egyénileg nem tudnám kiszolgálni az üzlettulajdonos által érkező folyamatos frissítési kéréseit. Ezt a felmerülő nehézséget úgy próbáltam megoldani, hogy a vállalkozó által is kényelmesen elérhetővé tegyem azokat a funkciókat, ami a webalkalmazás aktualizáltságát biztosítja. Továbbá a program képes a tulajdonos és a vásárló közötti közvetlen kapcsolat látszatát kialakítani a **chatbot**[1] funkció segítségével. Mivel ez egy egyszerűbb webalkalmazás, ezért nem egy teljesen egyénileg gondolkozó mesterséges intelligencia(MI)[2] alapú chatbotról esik szó, hanem egy adatbázisban tárolt előre legenerált válaszokból álló szöveges adathalmazról beszélhetünk, ami kulcsszavas keresés segítségével zajlik.

[1] egy szoftver alkalmazás, aminek a támogatásával közvetlen emberi kapcsolat helyett egy virtuális 'asszisztenssel' kommunikáljon.

[2] sokféle megközelítést találhatunk a definícióját illetően. Személy szerint azt gondolom, hogy az MI egy tudatos gondolkozásra alkalmas, emberi beavatkozás nélküli cselekvőképes létforma, amit a legtöbbször számítástechnikai eszközökhöz/gépekhez társítunk.

2. fejezet

Felhasználói dokumentáció

Lorem ipsum dolor sit amet $\mathbb{N}$, consectetur adipiscing elit. Duis nibh leo, dapibus in elementum nec, aliquet id sem. Suspendisse potenti. Nullam sit amet consectetur nibh. Donec scelerisque varius turpis at tincidunt. Cras a diam in mauris viverra vehicula. Vivamus mi odio, fermentum vel arcu efficitur, lacinia viverra nibh. Aliquam aliquam ante mi, vel pretium arcu dapibus eu. Nulla finibus ante vel arcu tincidunt, ut consectetur ligula finibus. Mauris mollis lectus sed ipsum bibendum, ac ultrices erat dictum. Suspendisse faucibus euismod lacinia $\mathbb{Z}$.

2.1. Felsorolások

Etiam vel odio ante. Etiam pulvinar nibh quis massa auctor congue. Pellentesque quis odio vitae sapien molestie vestibulum sit amet et quam. Pellentesque vel dui eget enim hendrerit finibus at sit amet libero. Quisque sollicitudin ultrices enim, nec porta magna imperdiet vitae. Cras condimentum nunc dui, eget molestie nunc accumsan vel.

- Fusce in aliquet neque, in pretium sem.
- Donec tincidunt tellus id lectus pretium fringilla.
- Nunc faucibus, erat pretium tempus tempor, tortor mi fringilla neque, ac congue ex dui vitae mauris.

Donec dapibus sodales ante, at scelerisque nunc laoreet sit amet. Mauris porttitor tincidunt neque, vel ullamcorper neque pulvinar et. Integer eu lorem euismod,

faucibus lectus sed, accumsan felis. Nunc ornare mi at augue vulputate, eu venenatis magna mollis. Nunc sed posuere dui, et varius nulla. Sed mollis nibh augue, eget scelerisque eros ornare nec.

1. Donec pretium et quam a cursus. Ut sollicitudin tempus urna et mollis.
2. Aliquam et aliquam turpis, sed fermentum mauris. Nulla eget ex diam.
3. Donec eget tellus pharetra, semper neque eget, rutrum diam Step 1.

Praesent porta, metus eget eleifend consequat, eros ligula eleifend ex, a pellentesque mi est vitae urna. Vivamus turpis nunc, iaculis non leo eget, mattis vulputate tellus. Maecenas rutrum eros sem, pharetra interdum nulla porttitor sit amet. In vitae viverra ante. Maecenas sit amet placerat orci, sed tincidunt velit. Vivamus mattis, enim vel suscipit elementum, quam odio venenatis elit[1], et mollis nulla nunc a risus. Praesent purus magna, tristique sed lacus sit amet, convallis malesuada magna.

Vestibulum venenatis malesuada enim, ac auctor erat vestibulum et. Phasellus id purus a leo suscipit accumsan.

Orci varius natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Nullam interdum rhoncus nisl, vel pharetra arcu euismod sagittis. Vestibulum ac turpis auctor, viverra turpis at, tempus tellus.

Morbi dignissim erat ut rutrum aliquet. Nulla eu rutrum urna. Integer non urna at mauris scelerisque rutrum sed non turpis.

2.1.1. Szoros térközű felsorolások

Phasellus ultricies, sapien sit amet ultricies placerat, velit purus viverra ligula, id consequat ipsum odio imperdiet enim:

1. Maecenas eget lobortis leo.
2. Donec eget libero enim.
3. In eu eros a eros lacinia maximus ullamcorper eget augue.

In quis turpis metus. Proin maximus nibh et massa eleifend, a feugiat augue porta. Sed eget est purus. Duis in placerat leo. Donec pharetra eros nec enim convallis:

[1] Phasellus faucibus varius purus, nec tristique enim porta vitae.

- Pellentesque odio lacus.
- Maximus ut nisl auctor.
- Sagittis vulputate lorem.

Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Sed lorem libero, dignissim vitae gravida a, ornare vitae est.

Cras maximus massa commodo pellentesque viverra.

Morbi sit amet ante risus. Aliquam nec sollicitudin mauris

Ut aliquam rhoncus sapien luctus viverra arcu iaculis posuere

2.2. Képek, ábrák

Aliquam vehicula luctus mi a pretium. Nulla quam neque, maximus nec velit in, aliquam mollis tortor. Aliquam erat volutpat. Curabitur vitae laoreet turpis. Integer id diam ligula. Nulla sodales purus id mi consequat, eu venenatis odio pharetra. Cras a arcu quam. Suspendisse augue risus, pulvinar a turpis et, commodo aliquet turpis. Nulla aliquam scelerisque mi eget pharetra. Mauris sed posuere elit, ac lobortis metus. Proin lacinia sit amet diam sed auctor. Nam viverra orci id sapien sollicitudin, a aliquam lacus suscipit, Figure 2.1:

2.1. ábra. Quisque ac tincidunt leo

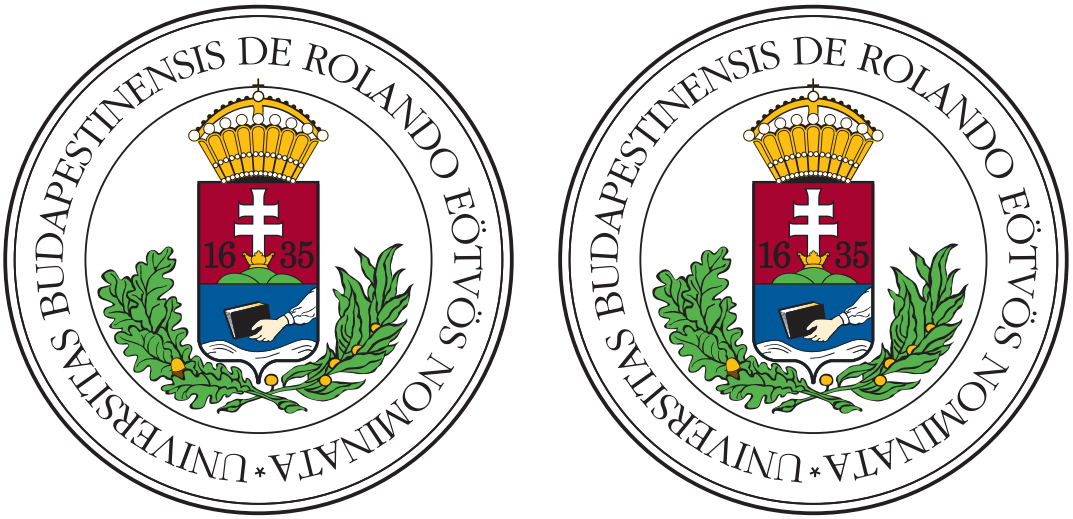
2.2.1. Képek szegélyezése

Ut aliquet nec neque eget fermentum. Cras volutpat tellus sed placerat elementum. Quisque neque dui, consectetur nec finibus eget, blandit id purus. Nam eget ipsum non nunc placerat interdum.

2.2. ábra. Quisque ac tincidunt leo

2.2.2. Képek csoportosítása

In non ipsum fermentum urna feugiat rutrum a at odio. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Nulla tincidunt mattis nisl id suscipit. Sed bibendum ac felis sed volutpat. Nam pharetra nisi nec facilisis faucibus. Aenean tristique nec libero non commodo. Nulla egestas laoreet tempus. Nunc eu aliquet nulla, quis vehicula dui. Proin ac risus sodales, gravida nisi vitae, efficitur neque, Figure 2.3:



(a) Vestibulum quis mattis urna

(b) Donec hendrerit quis dui sit amet venenatis

2.3. ábra. Aenean porttitor mi volutpat massa gravida

Nam et nunc eget elit tincidunt sollicitudin. Quisque ligula ipsum, tempor vitae tortor ut, commodo rhoncus diam. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Phasellus vehicula quam dui, eu convallis metus porta ac.

2.3. Táblázatok

Nam magna ex, euismod nec interdum sed, sagittis nec leo. Nam blandit massa bibendum mattis tristique. Phasellus tortor ligula, sodales a consectetur vitae, placerat vitae dolor. Aenean consequat in quam ac mollis.

Phasellus tortor	**Aenean consequat**
Sed malesuada	Aliquam aliquam velit in convallis ultrices.
Purus sagittis	Quisque lobortis eros vitae urna lacinia euismod.
Pellentesque	Curabitur ac lacus pellentesque, eleifend sem ut, placerat enim. Ut auctor tempor odio ut dapibus.

2.1. táblázat. Maecenas tincidunt non justo quis accumsan

2.3.1. Sorok és oszlopok egyesítése

Mauris a dapibus lectus. Vestibulum commodo nibh ante, ut maximus magna eleifend vel. Integer vehicula elit non lacus lacinia, vitae porttitor dolor ultrices. Vivamus gravida faucibus efficitur. Ut non erat quis arcu vehicula lacinia. Nulla felis mauris, laoreet sed malesuada in, euismod et lacus. Aenean at finibus ipsum. Pellentesque dignissim elit sit amet lacus congue vulputate.

Quisque	**Suspendisse**		**Aliquam**		**Vivamus**	
	Proin	Nunc	Proin	Nunc	Proin	Nunc
Leo	2,80 MB	100%	232 KB	8,09%	248 KB	8,64%
Vel	9,60 MB	100%	564 KB	5,74%	292 KB	2,97%
Auge	78,2 MB	100%	52,3 MB	66,88%	3,22 MB	4,12%

2.2. táblázat. Vivamus ac arcu fringilla, fermentum neque sed, interdum erat. Mauris bibendum mauris vitae enim mollis, et eleifend turpis aliquet.

2.3.2. Több oldalra átnyúló táblázatok

Nunc porta placerat leo, sit amet porttitor dui porta molestie. Aliquam at fermentum mi. Maecenas vitae lorem at leo tincidunt volutpat at nec tortor. Vivamus semper lacus eu diam laoreet congue. Vivamus in ipsum risus. Nulla ullamcorper finibus mauris non aliquet. Vivamus elementum rhoncus ex ut porttitor.

Praesent aliquam mauris enim	
Suspendisse potenti	*Lorem ipsum dolor sit amet*
Praesent	Nulla ultrices et libero sit amet fringilla. Nunc scelerisque ante tempus sapien placerat convallis.
Luctus	Integer hendrerit erat massa, non hendrerit risus convallis at. Curabitur ultrices, justo in imperdiet condimentum, neque tortor luctus enim, luctus posuere massa erat vitae nibh.
Egestas	Duis fermentum feugiat augue in blandit. Mauris a tempor felis. Pellentesque ultricies tristique dignissim. Pellentesque aliquam semper tristique. Nam nec egestas dolor. Vestibulum id elit quis enim fringilla tempor eu a mauris. Aliquam vitae lacus tellus. Phasellus mauris lectus, aliquam id leo eget, auctor dapibus magna. Fusce lacinia felis ac elit luctus luctus.
Dignissim	Praesent aliquam mauris enim, vestibulum posuere massa facilisis in. Suspendisse potenti. Nam quam purus, rutrum eu augue ut, varius vehicula tellus. Fusce dui diam, aliquet sit amet eros at, sollicitudin facilisis quam. Phasellus tempor metus vel augue gravida pretium. Proin aliquam aliquam blandit. Nulla id tempus mi. Fusce in aliquam tortor.
Pellentesque	Donec felis nibh, imperdiet a arcu non, vehicula gravida nibh. Quisque interdum sapien eu massa commodo, ac elementum felis faucibus.
Molestie	Cras ullamcorper tellus et auctor ultricies. Maecenas tincidunt euismod lectus nec venenatis. Suspendisse potenti. Pellentesque pretium nunc ut euismod cursus. Nam venenatis condimentum quam. Curabitur suscipit efficitur aliquet. Interdum et malesuada fames ac ante ipsum primis in faucibus.

Suspendisse potenti	*Lorem ipsum dolor sit amet*
Vivamus semper	In purus purus, faucibus eu libero vulputate, tristique sodales nunc. Nulla ut gravida dolor. Fusce vel pellentesque mi, vel efficitur eros. Nunc vitae elit tellus. Sed vestibulum auctor consequat.
Condimentum	Nulla scelerisque, leo et facilisis pretium, risus enim cursus turpis, eu suscipit ipsum ipsum in mauris. Praesent eget pulvinar ipsum, suscipit interdum nunc. Nam varius massa ut justo ullamcorper sollicitudin. Vivamus facilisis suscipit neque, eu fermentum risus. Ut at mi mauris.

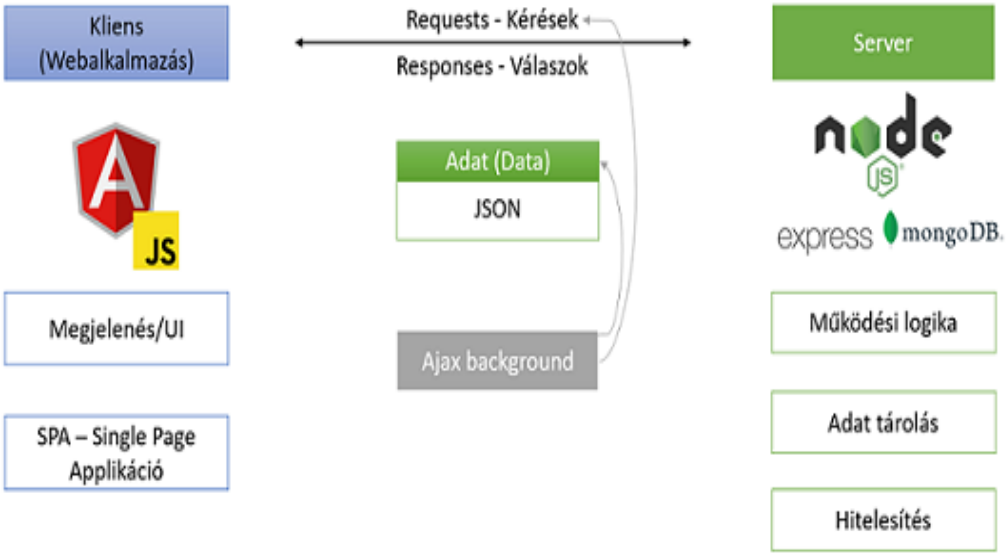
2.3. táblázat. Praesent ullamcorper consequat tellus ut eleifend

3. fejezet

Fejlesztői dokumentáció

3.1. Webalkalmazás specifikáció

Az alkalmazás fő célja egy olyan működő webáruház bemutatása, ami Angular keretrendszer segítségével készült. A webalkalmazás két főrészre osztható kliensoldali és szerveroldali (szakmai nyelven kifejezve: front-end és back-end) részre. A kliensoldal leglényegesebb feladata, hogy a felhasználó által is látott weboldalt megjelenítse grafikai UI/UX(rövidítés feloldása: User interface/User experience)dizájn implementálásával. Nevezetesen egy olyan rendszer, ami képes a felhasználó számára felületet és élményt biztosítani. Miközben a szerveroldal elsődleges feladata az alkalmazás úgynevezett business logikájának a megvalósítása. Ezen felül képes az adatok feldolgozására és hitelesítésére is. A következő ábrán szeretném reprezentálni milyen módon épül fel a webalkalmazás, továbbá megismertetni a két oldal kommunikációs kapcsolatát.



3.1. ábra. Az alkalmazás bemutatása

A 3.1-es ábrán látható a két oldal miképpen osztja meg az információkat egymás között. A front-end pontosabban mondva a kliensoldal Angular keretrendszerben TypeScript segítségével íródott. A front-end kommunikációja úgynevezett requestekkel más néven kérésekkel (json típusú adattovábbítással) a háttérben aszinkron módon történik amire a szerveroldal responsokkal egyszóval válaszokkal felel. A back-end Node.js szoftverrendszer alapú, ami Express segítségével íródott. Az adatok tárolásáért a MongoDB nevezetű adatbázis felel.

A következő blokkokban szeretném tételesen demonstrálni a fentebb említett front-end és back-end oldalakon használt módszerek alkalmazását és működését. Ezen felül szándékomban áll ismertetni az általam alkalmazott szoftverek technikai tulajdonságaikat.

3.2. Kliensoldalon használt technológiák bemutatása

3.3. Kliensoldal működése

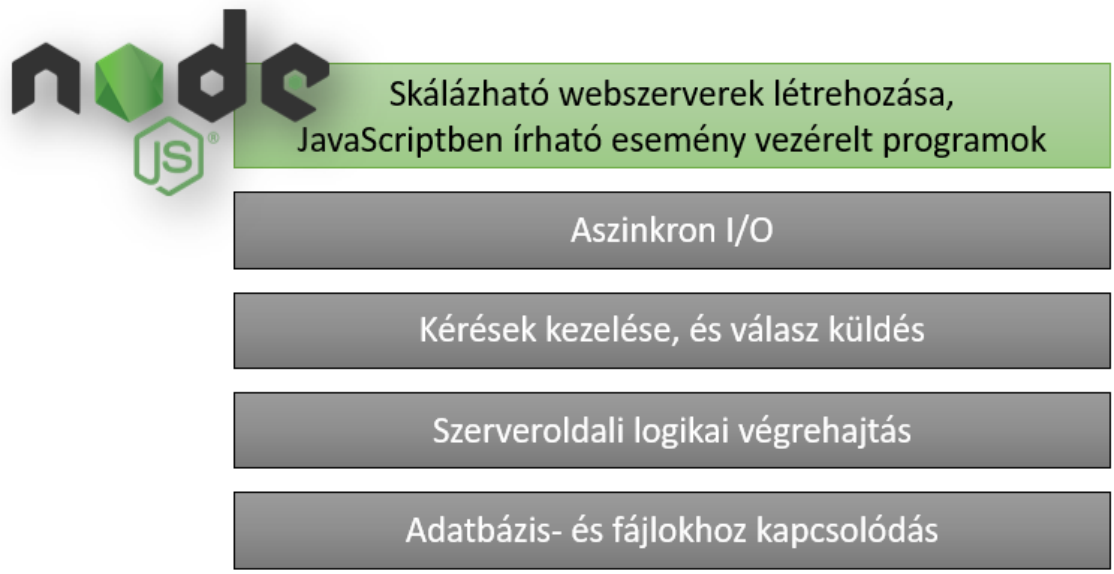
3.4. Szerveroldalon használt technológiák bemutatása

A szerveroldalon használt (3.1-es blokkban már említésre kerülő) módszerek kiválasztásuk előtt kulcsfontosságú szempontjuknak tartottam, hogy az Angular keretrendszerhez megfelelő kompatibilitással rendelkezzenek és alkalmazni tudjam őket a szakdolgozat elkészítése során. A következő technológiák mind olyan szoftverek, frameworkök vagy szerverek amikhez számtalan monográfia elérhető az interneten, ezzel támogatva a későbbiekben létrehozott projekteket. A következő felsorolásban összefoglalásképpen összegyűjtöttem az alkalmazásban fellelhető általam használt szoftvereket és verziószámukat:

- Node.js: 14.15.4
- Express.js: 4.17.1
- Mongoose: 6.0.12
- MongoDB Atlas

A fejezet további részében szeretném ismertetni a fentebb felsorolt technológiák jellegzetes tulajdonságaikat, főbb jellemzőiket további ábrák segítségével.

3.4.1. Node.js szoftverrendszer



3.2. ábra. NodeJS bemutatása

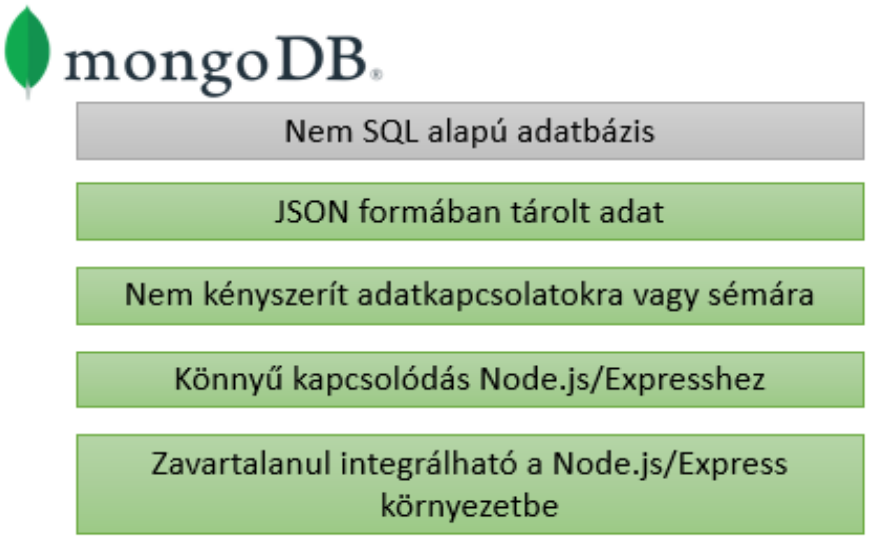
A webáruház back-end megírásánál 14.15.4-es verziójú Node.js szoftverrendszert használtam. A 3.2-es ábrán látható a Node.js bemutatása, ami összefoglalja a szoftverrendszer fontosabb jellemzőit. Az illusztráció első dobozában olvasható miszerint a Node.js skálázható webszerverek létrehozására alkalmas más szóval egy olyan rendszert tudunk létrehozni a támogatásával, ami több felhasználót képes egyidejűleg kiszolgálni. Ezenfelül JavaScript nyelv segítségével olyan programok írhatóak, amely a komponensek közötti esemény interakciókat tekinti alapul, mászóval eseményvezérelt programok megírására alkalmas (ilyen például egy egérkattintás vagy billentyű leütés). Folytatólag a Node.js aszinkron tulajdonságával lehetővé teszi, hogy a kliensoldalról érkező kérések várakozási sorrendbe kerüljenek, ennek következtében a kliensoldal tovább folytathatja a feladatát.

3.4.2. Express.js framework és Mongoose programozási könyvtár

3.3. ábra. Express bemutatása

A szerveroldal áttekinthetőbb és olvashatóbb kódírása érdekében a webáruház fejlesztése során az Express.js 4.17.1-es verzióját használtam. Az Express egy webes keretrendszer a Node.js nehézség nélküli használatára lett fejlesztve. A 3.3-as ábrán látható az Express attribútumainak ismertetése. Az illusztráción felvetettem, hogy az Express egy Middleware-típusú rendszer következésképpen lehetővé teszi a MongoDB adatbázis szoftverhez való zavartalan kapcsolódást a Mongoose 6.0.12 verziójával kiegészítve, ami egy JavaScript-ben írt objektum-orientált programozási könyvtár.

3.4.3. MongoDB adatbázisszerver

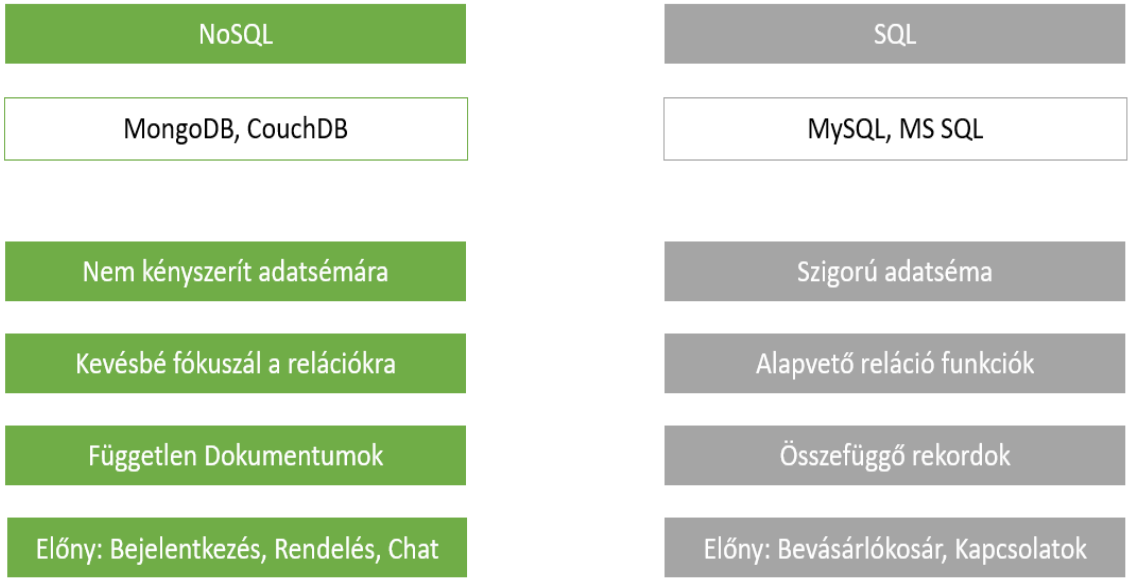


3.4. ábra. MongoDB bemutatása

A MongoDB egy nyílt forráskódú, NoSQL adatbázisszerverek közé sorolt szoftver. A NoSQL magyarán Nem SQL típusú adatbázisrendszert jelent. Jellemzően nem rekordokat és táblázatokat tárolnak mint az SQL típusú szerverek, hanem független dokumentumokat és gyűjteményeket archiválnak. Személyes véleményem szerint a 3.4-es illusztrációval alátámasztva egyik legfőbb pozitív tulajdonságának éreztem a alkalmazás írása során, hogy az adatok JSON formátumban képes tárolni. Következésképpen a request és response folyamatok egyszerűsített és gyors működését képes biztosítani, mindeközben lehetővé teszi a kliensoldalon megjelenítendő információk könnyebb feldolgozását.

3.4.4. NoSQL vs SQL adatbázisok összehasonlítása

A következő grafikai ábrán szándékozom röviden bemutatni és összehasonlítani a Nem SQL és az SQL alapú adatbázisokat jellegzetes tulajdonságaik szerint.



3.5. ábra. NoSQL vs SQL adatbázisok összehasonlítása

Mint a 3.5-ös ábrán megfigyelhető szempontok szerint egy webáruházban kezelt adatok tárolására a No SQL adatbázisok is kifejezetten alkalmasok. A NoSQL adatbázisszerverek jellemző tulajdonságai kulcsfontosságú szempontokkal szolgált a webáruház adatbázisának kiválasztásánál.

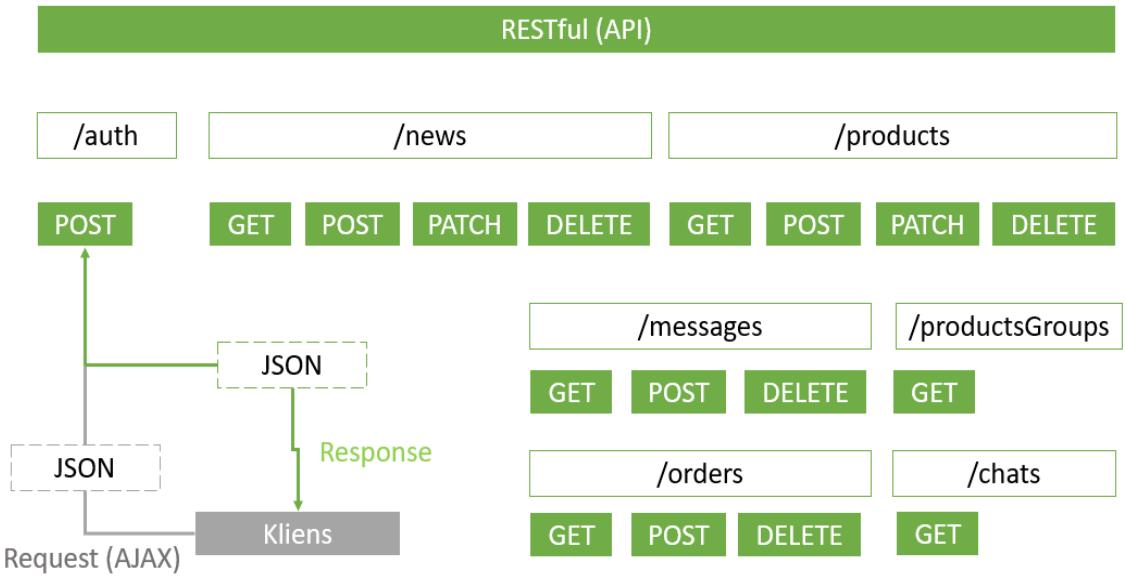
3.5. Szerveroldal működése

Ebben a fejezetben részletesen prezentálom az alkalmazás szerveroldali működését példának okáért milyen request és response hívások találhatóak a kódban, hogyan létesít kapcsolatot a webalkalmazás az adatbázissal...stb., valamint külön kitérek az admin oldalon található hitelesítésre alkalmazott technikára is.

3.5.1. RESTful API - Végpont tervek bemutatása

A webáruház megírása során RESTful API-t (feloldva: Representational State Transfer Application programming interface) magyarán reprezentáción alapuló állapotátvitel nevezetű architekturális módszert használtam. Az API-k segítségével a felhasználó számára elérhető a weboldalon számos interakció. Ilyen interakciónak számít például a bejelentkező felület.

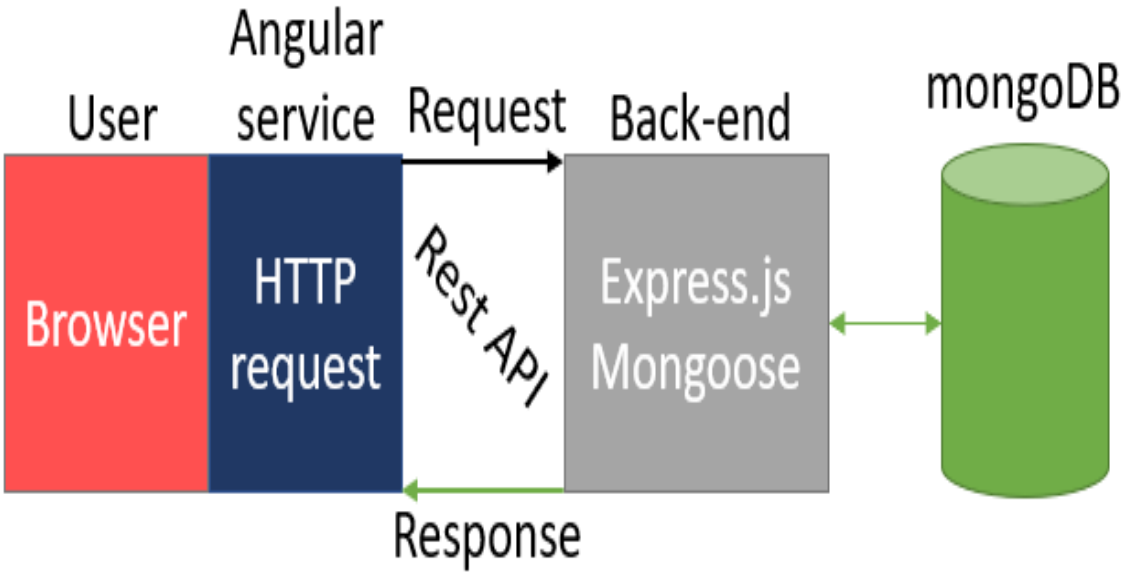
Az alábbi illusztráción szeretném bemutatni az alkalmazásban használt REST API hívásokat, ennek okán a 3.6-os ábrán láthatók a programban megírt végpontok.



3.6. ábra. Adatkezelés

A grafikán megfigyelhető hét különböző végpont ami az auth, hírek, termékek, termékcsoportok, üzenetek, chatek és rendeléseket fejezi ki. Az illusztrációt figyelemmel kísérve identifikálható, hogy nem minden végpont rendelkezik ugyan azokkal a kérésekkel. Szemléltetésképp vegyük figyelembe a hírekhez vonatkozó responsokat amik a GET, POST, PATCH és DELETE függvények, ezzel szemben a auth-hoz kizárólag POST függvény tartozik. Ennek kifejezetten egy oka van, mégpedig az, hogy a webáruház egyes adatait nem szükséges módosítani tudni kliens oldalról.

A REST API-t megismerve és ezt az információt felhasználva szemléltethető és az alább ábrán látható miképpen éri el a felhasználó által kezdeményezett kérés az adatbázist és hogyan kerül válaszra.



3.7. ábra. Kapcsolat a szerverrel

A fenti 3.7-es ábrával és az alább található alkalmazásban szereplő kódsorok segítségével szeretném bemutatni, hogy a felhasználó által indított kérés milyen sorrendben jut el az adatbázishoz és miképpen tér vissza hozzá.

Felhasználó interakcióba lép a felülettel (például: egy gombra kattintva 3.1-es forráskód), ennek következményeként meghívódik egy a gombhoz tartozó TypeScript nyelven írt függvény 3.2-es forráskód.

```
1  <form style="margin-top: 2rem;" [formGroup]="form" (submit)="
       onAddMessage()" *ngIf="!isLoading">
2  ....
3  <button class="btn-secondary" type="submit" [disabled]="!
       confirmed.checked">
4    {{'GENERIC.ACTION.SEND' | translate}}
5  </button>
```

3.1. forráskód. Felhasználói interakció - HTML fájl

```
1  onAddMessage(): void {
2    if(this.form.invalid) {
3      this.alertService.warn('ALERT.WARN.INVALID_FORM');
4      this.form.markAllAsTouched();
5      this.isLoading = false;
6      return;
7    }
8    this.isLoading = true;
```

```
    this.contactService.sendMessage(
      this.form.value.firstName,
      this.form.value.lastName,
      this.form.value.email,
      this.form.value.description,
      this.form.value.image,
    )
    this.isLoading = false
    this.form.reset();
  }
```

3.2. forráskód. Interakció függvényhívás - TypeScript fájl

Az előbb említett 3.2-es forráskódban szereplő függvény átadja az adatokat a kliensoldalon található Service fájl sendMessage() nevezetű függvényének. Ez a fájl tartalmazza a következő 3.3-as forráskódban szereplő kódsort.

```
sendMessage(
  firstName: string,
  lastName: string,
  email: string,
  description: string,
  image: File | string
){
  const messagesData = new FormData();
  messagesData.append("firstName", firstName);
  messagesData.append("lastName", lastName);
  messagesData.append("email", email);
  messagesData.append("description", description);
  messagesData.append("image", image, firstName);
  this.http.post<{message: string, messages: Messages}>
  (environment.apiUrl + "messages", messagesData)
    .subscribe(responseData => {
      const messages: Messages = {
        id: responseData.messages.id,
        firstName: firstName,
        lastName: lastName,
        email: email,
        description: description,
        imagePath: responseData.messages.imagePath,
      };
      this.messages.push(messages);
```

```
26          this.msgUpdate.next([...this.messages]);
27          this.alert.success('ALERT.SUCCESS.ADD');
28          this.router.navigate(["/contact"]);
29        }, error => {
30          this.alert.error(error.error.message);
31        });
32    }
```

3.3. forráskód. HttpClient POST request - Service TypeScript fájl

Ebben a kódrészletben látható, ahogyan a kliens oldal HttpClient Angular package POST request segítségével a megadott útvonalon küld egy REST API kérést a szerveroldal felé.

A szerveroldal fogadja ezt a kérést és továbbítja a MongoDB felé. Az alábbi 3.4-es forráskódban Express - JavaScript nyelv segítségével megírt kódrészletben ez szerepel.

```
1   exports.postMessages = (req, res, next) => {
2     const url = req.protocol + "://" + req.get("host");
3     const msg = new Messages({
4       firstName: req.body.firstName,
5       lastName: req.body.lastName,
6       email: req.body.email,
7       description: req.body.description,
8       imagePath: url + "/images/messages/" + req.file.filename,
9     });
10    msg.save().then(result => {
11      res.status(201).json({
12        message: "Message added successfully",
13        messages: {
14          ...result,
15          id: result._id
16        }
17      });
18    });
19  }
```

3.4. forráskód. Express POST végpont - JavaScript

Kliensoldalon és Szerveroldalon egyaránt látható a felhasználó által elindított kérés státuszát tartalmazó információ. A front-end-en egy úgynevezett AlertService

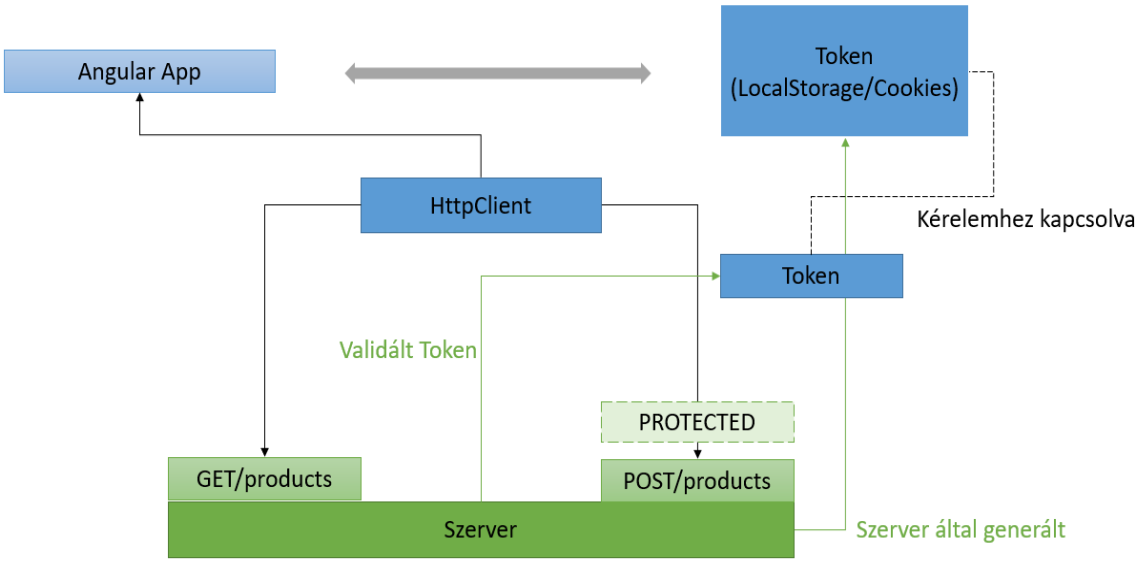
componens segítségével kap visszaigazolást az elindított kérés befejezéséről, míg a back-end-en az adatbázis felé elküldött kérés kódrészletében található ez az információ.

A webalkalmazásban szereplő további REST API hívásokat tartalmazó példakódsorokat a 3.7-es Forráskódok 3.7.2-es Szerveroldali forráskódok nevezetű alfejezetben találhatóak.

3.5.2. Hitelesítés - Admin felület védelme

Az alkalmazás megírása során akaratlagosan egy olyan webáruház létrehozása volt a végcélom, ami időszerű adatok kezelését tudja biztosítani. Következésképpen egy olyan felület megvalósítása gyakorlatban is, amely nem igényel programozón keresztüli beavatkozást. Ennek okán a szakdolgozat tartalmaz egy adminisztrációs oldalt, amely biztosítja a webalkalmazásban dinamikusan megjelenő adatok aktualitását, továbbá a leadott rendelések vásárlók által elküldött üzenetek megjelenítésére is szolgál. Kifejezetten ennek a blokk védelmében került bele külön hitelesítési felület.

Az alább található 3.8-as grafikán ez az engedélyezési folyamat elméleti működése látható.



3.8. ábra. Hitelesítés

Az illusztráción ábrázolt hitelesítési folyamat a következőképpen történik. Az alkalmazás front-end-ről érkező GET/products kérésre engedélyezés nélkül kap választ

a szervertől, mivel a webáruházban bejelentkezés hiányában is hozzáférhetővé kell tenni ezt az információt. Ezzel szemben a POST/products kérés nem következhet be hitelesítés nélkül. Tehát bejelentkezés során a szerver generál egy úgynevezett tokent és LocalStorage-ba elmenti, amit az új termék hozzáadás kérésnél ellenőriz. Az alkalmazásban ez a hitelesítési funkciót következőképpen valósul meg:

Ennek a funkciónak létrehozásánál két kiegészítő package-et úgynevezett bcryptjs-t és jsonwebtoken-t használja a programkód. Az előbbi lehetőséget nyújt bizonyos adatok titkosítására (ilyen adat például a belépési jelszó), az utóbbi pedig bizonyos REST API kérések érvényesítésére alkalmas. Mikor létrehozásra kerül egy új felhasználó a bcrypt package egy úgynevezett hash metódusát hívja meg a program, aminek segítségével az új felhasználó jelszava titkosítva kerül be az adatbázisba. Ennek oka az, hogy ha véletlenül sikerül egy idegennek belépnie az adatbázisba, akkor ne tudja megszerezni az adott felhasználók jelszavát. Viszont, ha titkosítva kerül be az adat, a későbbiekben belépésnél nem lehet ellenőrizni, hogy a megfelelő jelszó került-e beírásra. Erre a problémára szolgál megoldásképp a bcryptjs compare metódusának használata, aminek segítségével összehasonlításra kerül az adott felhasználó adatbázisban szereplő jelszava és a begépelt jelszó hashelt változata, mivel ha pont a megfelelő adat kerül beírásra akkor a titkosított információnak megegyezőnek kell lennie az adatbázisban található jelszóval. Sikeres bejelentkezésnél a szerveroldal generál egy tokent a jsonwebtoken package segítségével és minden olyan kérésnél amihez szükséges autentikáció, ellenőrzésre kerül ennek az érvényes tokennek a létezése. A programkódban generált tokenek hitelessége egy óráig él.

Mindent összevetve az adminisztrációs felület védve van az illetéktelen felhasználók belépésétől és bizonyos funkciók végrehajtásától. A hitelesítési folyamatot leíró példakódok a 3.7-es Forráskódok, 3.7.2-es Szerveroldali forráskódok nevezetű alfejezetben találhatóak.

3.6. Fogalomtár, megjegyzések

1. Definíció. Mauris tristique sollicitudin ultrices. Etiam tristique quam sit amet metus dictum imperdiet. Nunc id lorem sed nisl pulvinar aliquet vitae quis arcu. Morbi iaculis eleifend porttitor.

Maecenas rutrum eros sem, pharetra interdum nulla porttitor sit amet. In vitae viverra ante. Maecenas sit amet placerat orci, sed tincidunt velit. Vivamus mattis, enim vel suscipit elementum, quam odio venenatis elit, et mollis nulla nunc a risus. Praesent purus magna, tristique sed lacus sit amet, convallis malesuada magna. Phasellus faucibus varius purus, nec tristique enim porta vitae.

1. Tétel. *Nulla finibus ante vel arcu tincidunt, ut consectetur ligula finibus. Mauris mollis lectus sed ipsum bibendum, ac ultrices erat dictum. Suspendisse faucibus euismod lacinia. Etiam vel odio ante.*

Bizonyítás. Etiam pulvinar nibh quis massa auctor congue. Pellentesque quis odio vitae sapien molestie vestibulum sit amet et quam. Pellentesque vel dui eget enim hendrerit finibus at sit amet libero. Quisque sollicitudin ultrices enim, nec porta magna imperdiet vitae. Cras condimentum nunc dui. □

Donec dapibus sodales ante, at scelerisque nunc laoreet sit amet. Mauris porttitor tincidunt neque, vel ullamcorper neque pulvinar et. Integer eu lorem euismod, faucibus lectus sed, accumsan felis.

Emlékeztető. Nunc ornare mi at augue vulputate, eu venenatis magna mollis. Nunc sed posuere dui, et varius nulla. Sed mollis nibh augue, eget scelerisque eros ornare nec. Praesent porta, metus eget eleifend consequat, eros ligula eleifend ex, a pellentesque mi est vitae urna. Vivamus turpis nunc, iaculis non leo eget, mattis vulputate tellus.

Fusce in aliquet neque, in pretium sem. Donec tincidunt tellus id lectus pretium fringilla. Nunc faucibus, erat pretium tempus tempor, tortor mi fringilla neque, ac congue ex dui vitae mauris. Donec pretium et quam a cursus.

Megjegyzés. Aliquam vehicula luctus mi a pretium. Nulla quam neque, maximus nec velit in, aliquam mollis tortor. Aliquam erat volutpat. Curabitur vitae laoreet turpis. Integer id diam ligula.

Ut sollicitudin tempus urna et mollis. Aliquam et aliquam turpis, sed fermentum mauris. Nulla eget ex diam. Donec eget tellus pharetra, semper neque eget, rutrum diam.

3.7. Forráskódok

3.7.1. Kliensoldali forráskódok

Nulla sodales purus id mi consequat, eu venenatis odio pharetra. Cras a arcu quam. Suspendisse augue risus, pulvinar a turpis et, commodo aliquet turpis. Nulla aliquam scelerisque mi eget pharetra. Mauris sed posuere elit, ac lobortis metus. Proin lacinia sit amet diam sed auctor.

3.7.2. Szerveroldali forráskódok

Alább található forráskódok a 3.5.1-es fejezetben megemlített JavaScriptben írt egy-egy példa request végpontokat tartalmaz.

POST/news 3.5 és a visszaérkező json fájl 3.6:

```
exports.postNews = (req, res, next) => {
  const url = req.protocol + "://" + req.get("host");
  const news = new News({
    title: req.body.title,
    description: req.body.description,
    imagePath: url + "/images/news/" + req.file.filename,
    startDate: req.body.startDate,
    endDate: req.body.endDate,
  });
  news.save().then(result => {
    res.status(201).json({
      message: "News added successfully",
      news: {
        ...result,
        id: result._id
      }
    });
  });
}
```

3.5. forráskód. POST/news végpont

```
{
  "_id":
  {
    "$oid":"618e7a2edd33576f6e96b11c"
```

```
5   },
6   "title":"Teszt",
7   "description":"Teszt szoveg",
8   "imagePath":"http://localhost:3000/images/news/teszt
        -1636727342360.jpg",
9   "startDate":
10  {
11    "$date":
12    {
13      "$numberLong":"1637276400000"
14    }
15  },
16  "endDate":
17  {
18    "$date":
19    {
20      "$numberLong":"1637881200000"
21    }
22  }
```

3.6. forráskód. POST/news JSON

Az alábbi forráskódok a 3.5.2-es alfejezetben kifejtett hitelesítési funkció kódbeli megvalósítása olvasható. Ezen programkódok részben Maximilian Schwarzmüller által írt kód felhasználásával történt.

3.7.3. Algoritmusok

A general Interval Branch and Bound algorithm is shown in Algorithm 1. One of the following selection rules is applied in Step 3.
Példa forrása: Acta Cybernetica (ez egy link).

1. Algoritmus A general interval B&B algorithm

Funct IBB(S, f)

Set the working list $\mathcal{L}_W := \{S\}$ and the final list $\mathcal{L}_Q := \{\}$
while ($\mathcal{L}_W \neq \emptyset$) **do**
    Select an interval X from $\mathcal{L}_W$ ▷ Selection rule
    Compute $lbf(X)$ ▷ Bounding rule
    if X cannot be eliminated **then** ▷ Elimination rule
        Divide X into $X^j,\ j = 1, \ldots, p$, subintervals ▷ Division rule
        for $j = 1, \ldots, p$ **do**
            if X^j satisfies the termination criterion **then** ▷ Termination rule
                Store X^j in $\mathcal{L}_W$
            else
                Store X^j in $\mathcal{L}_W$
            end if
        end for
    end if
end while
return $\mathcal{L}_Q$

3.8. Tesztesetek

3.9. Továbbfejlesztési lehetőségek

4. fejezet

Összegzés

Lorem ipsum dolor sit amet, consectetur adipiscing elit. In eu egestas mauris. Quisque nisl elit, varius in erat eu, dictum commodo lorem. Sed commodo libero et sem laoreet consectetur. Fusce ligula arcu, vestibulum et sodales vel, venenatis at velit. Aliquam erat volutpat. Proin condimentum accumsan velit id hendrerit. Cras egestas arcu quis felis placerat, ut sodales velit malesuada. Maecenas et turpis eu turpis placerat euismod. Maecenas a urna viverra, scelerisque nibh ut, malesuada ex.

Aliquam suscipit dignissim tempor. Praesent tortor libero, feugiat et tellus porttitor, malesuada eleifend felis. Orci varius natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Nullam eleifend imperdiet lorem, sit amet imperdiet metus pellentesque vitae. Donec nec ligula urna. Aliquam bibendum tempor diam, sed lacinia eros dapibus id. Donec sed vehicula turpis. Aliquam hendrerit sed nulla vitae convallis. Etiam libero quam, pharetra ac est nec, sodales placerat augue. Praesent eu consequat purus.

A. függelék

Szimulációs eredmények

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque facilisis in nibh auctor molestie. Donec porta tortor mauris. Cras in lacus in purus ultricies blandit. Proin dolor erat, pulvinar posuere orci ac, eleifend ultrices libero. Donec elementum et elit a ullamcorper. Nunc tincidunt, lorem et consectetur tincidunt, ante sapien scelerisque neque, eu bibendum felis augue non est. Maecenas nibh arcu, ultrices et libero id, egestas tempus mauris. Etiam iaculis dui nec augue venenatis, fermentum posuere justo congue. Nullam sit amet porttitor sem, at porttitor augue. Proin bibendum justo at ornare efficitur. Donec tempor turpis ligula, vitae viverra felis finibus eu. Curabitur sed libero ac urna condimentum gravida. Donec tincidunt neque sit amet neque luctus auctor vel eget tortor. Integer dignissim, urna ut lobortis volutpat, justo nunc convallis diam, sit amet vulputate erat eros eu velit. Mauris porttitor dictum ante, commodo facilisis ex suscipit sed.

Sed egestas dapibus nisl, vitae fringilla justo. Donec eget condimentum lectus, molestie mattis nunc. Nulla ac faucibus dui. Nullam a congue erat. Ut accumsan sed sapien quis porttitor. Ut pellentesque, est ac posuere pulvinar, tortor mauris fermentum nulla, sit amet fringilla sapien sapien quis velit. Integer accumsan placerat lorem, eu aliquam urna consectetur eget. In ligula orci, dignissim sed consequat ac, porta at metus. Phasellus ipsum tellus, molestie ut lacus tempus, rutrum convallis elit. Suspendisse arcu orci, luctus vitae ultricies quis, bibendum sed elit. Vivamus at sem maximus leo placerat gravida semper vel mi. Etiam hendrerit sed massa ut lacinia. Morbi varius libero odio, sit amet auctor nunc interdum sit amet.

Aenean non mauris accumsan, rutrum nisi non, porttitor enim. Maecenas vel tortor ex. Proin vulputate tellus luctus egestas fermentum. In nec lobortis risus,

sit amet tincidunt purus. Nam id turpis venenatis, vehicula nisl sed, ultricies nibh. Suspendisse in libero nec nisi tempor vestibulum. Integer eu dui congue enim venenatis lobortis. Donec sed elementum nunc. Nulla facilisi. Maecenas cursus id lorem et finibus. Sed fermentum molestie erat, nec tempor lorem facilisis cursus. In vel nulla id orci fringilla facilisis. Cras non bibendum odio, ac vestibulum ex. Donec turpis urna, tincidunt ut mi eu, finibus facilisis lorem. Praesent posuere nisl nec dui accumsan, sed interdum odio malesuada.

Ábrák jegyzéke

Táblázatok jegyzéke

Algoritmusjegyzék

Forráskódjegyzék