Matematikai és Informatikai Intézet

Határidő napló webes és androidos platformon

Készítette
Kertész Zoltán
Programtervező informatikus BSc

Témavezető
Dr. Tajti Tibor Gábor
adjuntus

Eger, 2022

Tartalomjegyzék

Bevezetés

A tanulmányaim alatt lehetőségem volt többféle programozási nyelv megismerésére. A szerver és a kliens oldali programozási nyelvek között is találtam olyat, ami elnyerte a tetszésemet.

A mobil applikáció fejlesztéssel először egy beadandó feladatban találkoztam, majd később az ehhez tartozó órákat is elvégeztem, hogy elmélyítsem a tudásomat. A Java programozási nyelv elsajátítása nem okozott nagy kihívást köszönhetően a tanáraimnak.

A weboldal fejlesztési ismeretek úgy gondolom, hogy manapság egy alapvető elvárás ezért több erre szolgáló programozási nyelvet is tanultam, de ezek közül az PHP volt számomra a legkedveltebb. A választék igen nagy a PHP alapú keretrendszerek között, de számomra a Laravel lett a preferált.

Ezért az lett a célom, hogy szakdolgozatomban ezeket a technológiákat használjam. Ezek felhasználásra rengeteg terv volt a fejemben, mire kialakult a jelenlegi projekt.

Véleményem szerint az emberek manapság inkább különböző applikációkat használnak, hogy a teendőiket nyilvántartsák, mint a hagyományos papír alapú noteszt. Viszont csak a mobilon tárolni a fontos feljegyzéseinket veszélyes, mivel az egy sérülékeny eszköz. A dolgozatom lényege, hogy egy olyan alkalmazást hozzak létre, amiben a felhasználók el tudják tárolni az eseményeiket és el is érjék azokat távolról is, platform függetlenül. Mivel az okos telefonok rendelkeznek beépített böngészővel így elég lenne, a probléma megoldásához csak a webes alkalmazás is, de szívesebben használják a felhasználók a kliens programokat.

1. fejezet

Az alkalmazás bemutatása

A szakdolgozatom projektje két részből tevődik össze. Az első rész egy webes alkalmazás a második pedig egy mobil applikáció. A fő összetevője a webes felület, mivel Androidon a projekt nem minden funkciója érhető el.

A webes és az androidos alkalmazás tekintetében is törekedtem egy minimalista, mégis modern dizájn kialakításra, ami illeszkedik a világos és sötét témához is.

1.1. Webes alkalmazás

A projekt fő eleme a böngészőből elérhető alkalmazás. Itt került kialakításra a regisztráció és egy rövid ismertető a programról, hogy milyen feladatok, problémák megoldására terveztem. Ezek az oldalak elérhetőek a látogatók számára regisztráció nélkül.

A felhasználónak regisztrációkor meg kell adnia a nevét, email címét és jelszavát. Ha a regisztráció sikeres egy levelet küld ki az alkalmazás a megadott címre ahol a regisztrációkor kitöltött név jelenik meg és egy gomb ami átirányítja a felhasználót a webes felületre. Bejelentkezés után a felhasználó korlátozás nélkül tudja használni az alkalmazás nyújtotta lehetőségeket.

A navigációs részben a legördülő menüben az események alatt lehetőség van új esemény hozzáadására, a meglévők kilistázására annak megfelelően, hogy melyik csoportba tartozókat szeretnénk megjeleníteni. Ilyen kategóriák az aktív, teljesített és lejárt események.

Ahhoz, hogy a használó új eseményt tudjon hozzáadni kötelező annak nevet, kezdési és befejezési dátumot adni. Mentés közben ezeken az adatokon ellenőrzés fut le, hogy a kitöltés a szabályoknak megfelelő legyen. Ilyen előírás, hogy a név és dátumok ki legyenek töltve. A dátumokra másik szabály is vonatkozik, mégpedig, hogy a kezdési időpont ne lehessen korábban, mint az aktuális dátum és a befejezési idő nem lehet korábban, mint a kezdési. Ha ezeknek a bevitt értékek megfelelnek akkor megtörténik az eseménynek a mentése és közben az alkalmazás levélben értesíti a felhasználót, amiben leírja neki az eseményhez tartozó elnevezést, leírást, kezdési és befejezési dátumot.

Az feljegyzések közötti kilistázás első eleme az „Aktív események”. Az alkalmazásomban az számít futónak, ahol a befejezési időpont nem korábbi, mint az éppen aktuális dátum és az adatbázisban a „complete” attribútumban 0 szerepel.

A menüben a következő választható lehetőség a „Lejárt események” listája. Itt van lehetősége a felhasználónak megtekintenie azokat az eseményeket amiket nem teljesített, de már nem is aktívak. Lejárt eseménynek a rendszerben az számít ahol a befejezési dátum korábbi, mint az aktuális dátum és a „complete” oszlop értéke szintén 0.

A fenti két eseményben az a közös, hogy a kliensnek lehetőse van a módosításra, teljesítésre és a törlésre is. Módosításkor a hozzáadási szabályok érvényesek az adatokra.

A legördülő menü utolsó eleme pedig a „Teljesített események”. Itt lehet megtalálni azokat az eseményeket ahol a „complete” oszlopnak 1 az értéke. A megjelenő rekordok a módosítási dátumuk alapján csökkenő sorrendben kerülnek megjelenítésre. A teljesített események fontos különbsége, hogy a módosításra már nincsen lehetősége a felhasználónak, csak a törlésre.

A következő választható menü elem a „Profil”. Itt egy olyan oldal jelenik meg, ahol a bejelentkezett felhasználó adati kerülnek kiíratásra. Az információk alatt egy legördülő menü található amiben a profilon elvégezhető műveleteket érhetjük el.

A „Profil módosítás”-t választva megjelennek a felhasználó adatai. Amennyiben egy adatot nem szeretnénk módosítani úgy azt nem kell átírni. A mentésre kattintva az adatokat ellenőrízük, ahol a következő szabályoknak meg kell felelni: a név mező nem lehet üres, az e-mail cím nem lehet üres és tartalmaznia kell „@” írásjelet.

Lehetőség van jelszó módosításra is, amit a következő menüpont biztosít. Jelszó módosításkor egyértelmű, hogy az üres mezők esetén hibával térünk vissza. Ezen kívül az új jelszót kétszer kell megadni és vizsgáljuk, hogy ugyan azok-e a jelszavak. Ezt követően az új jelszónak beírt karakterláncot kódolva eltároljuk, kijelentkeztetésre kerül a felhasználó és át lesz irányítva a bejelentkező oldalra, ahol már az új jelszóval kell bejelentkeznie.

Az utolsó funkció amit a felhasználó elér ebben a menüben az a profiljának a törlése. Hogy ez sikeres legyen meg kell adni kétszer a jelenlegi jelszót. Ha a törlés sikeres a felhasználó az applikáció főoldalára lesz irányítva, amennyiben sikertelen egy hiba üzenettel küldjük vissza a törlési oldalra.

A navigációs fül utolsó lehetőse a „Kijelentkezés”. A bejelentkezett felhasználó itt ki tud lépni az alkalmazásból és a kezdőoldalra kerül.

1.2. Androidos alkalmazás

A kliens program Androidos telefonokra készült alkalmazás ami REST API kapcsolaton keresztül kommunikál a webes szoftverrel. Az alkalmazás nem tárol lokálisan adatot, ezzel került megoldásra az a probléma, hogy esetleges meghibásodás esetén adat vesztés

történne.

Az alkalmazás első megnyitása után egy bejelentkezési felület fogadja a felhasználót ahol meg kell adni az e-mail címét és jelszavát. Ezen két adat ellenőrzése két helyen valósul meg. Először az alkalmazás megvizsgálja, hogy nem-e üresek a mezők, amennyiben nem írtunk bele adatot hibaüzenetet kapunk. Ha kivannak töltve a mezők egy REST kérést küldünk. Amennyiben a válasz nem a felhasználó adatai egy szöveg buborék jelenik meg, hogy hibás a bevitt adat, máskülönben sikeres a bejelentkezés és a kezdőképernyő jelenik meg.

A főoldalon lehetőség van új eseményt hozzáadni vagy kilistázni az aktív, lejárt vagy teljesített feljegyzéseket. Ezen kívül található még egy felugró menü ahol minden funkciót megtalálunk amit elérhetünk az alkalmazásban. A menüben olyan pontokat találhatunk ami az esemény hozzáadás, listázás típusonként, feljegyzés kezelése, profil szerkesztése és a kijelentkezés.

Az esemény hozzáadása több helyről is elérhető, mivel ez egy alap funkció, ezért kézközelben kell lennie. Amikor megnyílik a hozzáadásért felelős nézet a már webes felületen megszokott adatok bevitelére van lehetőség. Dátumot egy felugró ablak segítségével lehet beállítani, majd megjelenítjük a kiválasztott időpontot.

A főoldalon a bejelentkezett felhasználót egy üdvözlő üzenet fogadja, ahol a regisztrációkor megadott név szerepel. Ezen kívül elérhetjük a hozzáadás funkciót és esemény csoportonként a listázást.

A felugró menüben is elérhetőek a fent említett funkciók és kiegészülnek az esemény kezelésével, profil megjelenítéssel és a kijelentkezéssel.

A csoportonként megjelenítendő eseményeknél a felhasználó a következő adatokat látja: az esemény azonosítója, neve, leírása, kezdeti és befejezési dátum. A webes felülethez hasonlóan itt is kártyaként jelennek meg az adatok.

Az „Esemény kezelése” menüpontban van lehetőség módosítani, késznek jelölni és törölni az adott eseményt. Először a felhasználónak le kell ellenőriznie, hogy az esemény létezik-e, ehhez meg kell adnia az azonosítóját. Amennyiben nem létezik egy felugró szöveg buborék jelenik meg, hogy helytelen az azonosító. Ha sikeres volt az ellenőrzés megjelenik a kiválasztott azonosítóhoz tartozó adatok. Mentés esetén a már webes felületen ismertetett szabályok kerülnek ellenőrzésre. Amennyiben üresen hagyja a felhasználó a nevet vagy a dátumok közül valamelyiket egy hiba üzenet kerül kiírásra ahol megjelenítésre kerül, hogy mely adatokat kell leellenőrizni.

A „Profil” menüpontot választva megjelenik az aktuálisan bejelentkezett felhasználó adatai amit módosítani lehet. A sikeres módosítás után az alkalmazás egy szöveg buborékban tájékoztatja a felhasználót a sikerességről.

A felhasználói adatokat bejelentkezés után megjegyzi a kliens program így nem kell minden megnyitáskor belépni csak, ha a menüben kijelentkezünk az alkalmazásból.

2. fejezet

Fejlesztői környezet

2.1. Web szerver

Az alkalmazások működéséhez szükséges egy web szerver. Ehhez több lehetőség is van a fejlesztők részére. Virtualizálhatunk egy Linux alapú szervert ahol kialakítjuk a LAMP (Linux, Apache, PHP, Myadmin) környezetet. Létrehozhatjuk a programunkat Dockerben, ami egy virtuális szervernek felel meg, vagy használhatunk olyan segéd programot, mint a XAMPP vagy a WAMPP.

Személy szerint nekem sokkal jobban bevált a XAMPP, így dolgozatom fejlesztése alatt is azt használtam, mivel egyszerű a telepítése és a kezelhetősége. Mindemellett lehetőség van konfigurálni azt is, hogy milyen adatbázist szeretnénk használni.

2.2. Kód szerkesztő

Ha már van környezetünk ahol majd a webes projektünk futtathatóvá válik keresnünk kell egy olyan szöveg szerkesztőt amit használni fogunk. Program fejlesztésre több ilyen eszköz is létezik, ami segíti a programozó munkáját szöveg kiemeléssel, kód kiegészítéssel és szintaxis ellenőrzéssel. Számomra a legjobban bevált ilyen program a „Visual Studio Code" ami ingyenes szövegszerkesztő és különböző bővítményekkel lehet kényelmesebbé tenni a használatát.

2.3. Android alkalmazáshoz Adroid Studio

Androidos alkalmazás fejlesztésére többféle eszköz áll a rendelkezésünkre. Ilyen lehetőség lehet az „Eclipse" vagy esetleg a „RAD Studio". Számomra a leginkább preferált fejlesztői környezet az „Android Studio". Azért tartom jobbnak a többinél, mivel nem csak mobil eszközre lehet vele alkalmazást írni, hanem „Wear OS"-es órára és TV-re is. A másik előny, hogy a felhasználói kezelő felület kialakításához biztosít egy tervezőt

ahol az elemeket fogd és vidd (drag and drop) módszerrel lehet elhelyezni a megjelenítő felületre.

Az „Android Studio”-ba olyan fontos funkciók vannak beépítve, mint az AVD menedzser (manager), ami egy emulátor, ahol virtuális eszközöket tudunk létrehozni, hogy tesztelni tudjuk az alkalmazásunkat. A másik fontos eszköz ami implementálva van az ADB (Android Debug Bridge) ami a virtuális vagy fizikai eszközünk között kommunikál a számítógéppel így figyelve az alkalmazásunk működését és hiba esetén megkönnyíti a hiba keresést.

2.4. Postman a REST API-hoz

A REST API kapcsolat teszteléséhez szükség volt egy kliens programra, ahol lelehet ellenőrizni, hogy a kérés sikeres-e és a válasz megfelelő-e. Ehhez a „Postman” programot választottam, mivel kliens program, egyszerű a használata ezért nem csak egy személy vehet részt a tesztelésben, mindezek mellett lehetőség van a teszteket kollekcióba elmenteni amiből későbbiekben automatizált teszt is futtatható. A szoftver nagyon sok API szabványt és formátumot támogat köztük a „JSON”-t is.

3. fejezet

Az alkalmazás felépítése

3.1. Miért PHP és Laravel?

A szerver oldali programozási nyelvek között még mindig vezető helyen szerepel a PHP, köszönhetően annak, hogy széles körben használt, egyszerű és könnyen tanulható nyelv. A másik vezető nyelv a JavaScript amit leginkább „SPA" (Single Page Application / Egyoldalas alkalmazások) fejlesztésére használnak.

Számomra a PHP azért lett kedvesebb, mivel széleskörűen használható, gyors, rugalmas és jól dokumentált.

Mivel tisztán PHP kódot már kevés helyen használnak, leginkább kis projektek esetén így érdemes keresni egy keretrendszert. Nagyon sok lehetőség van az interneten amit használhatunk, de talán a legelterjedtebb ilyen a Laravel.

A Laravel több szempontból is ideális választás webes alkalmazásokhoz, mivel megkönnyíti az olyan általános feladatokat, mint az útvonalválasztás (routing) és a hitelesítés. A keretrendszer követi az MVC (Model View Controller / Modell Nézet Vezérlő) tervezési mintát, ezért hatékonyabb, biztonságosabb és a projekt átlátható lesz.

Szinte minden webes alkalmazás kommunikál valamilyen adatbázissal. A Laravel ezt a kommunikációt teszi egyszerűbbé a Query Builder (lekérdezéskészítő) és az ORM (Object Relational Mapping / Objektum-relációs leképezés) használatával.

3.2. Sass és CSS

A CSS állományok gondoskodnak a frontend dizájnjáról. Ez sajnos sok hiba lehetőséggel jár, mivel egyetlen nagy fájlban tároljuk a megjelenésre vonatkozó információkat. Ennek eredménye, hogy nagy projektben strukturálatlanná válik az állomány aminek rendszerezése igen nagy feladat lenne. Erre jó mód lenne külön állományban tárolni az egyes oldalakra vonatkozó CSS információkat, de ez több kérés lenne a szerver felé.

Ennek a problémának a megoldására léteznek olyan előfeldolgozók (pre-processor),

mint a Sass, ahol lehetőség van több rész állományt létrehozni ezzel javítva a struktúrát. Az előfeldolgozó végül pedig egyetlen fájlba egyesíti a részeket és nem növeli a kérések számát.

3.3. Bootstrap

A Bootstrap egy olyan nyílt forráskódú CSS keretrendszer ami megkönnyíti a frontend programozást az előre definiált komponensekkel. Ez a technológia minimális erőfeszítést igényel a fejlesztő részéről, hogy implementálja saját projektjében. Az előre definiált komponensek tartalmaznak HTML, JavaScript és CSS elemeket, melyeket a későbbiekben könnyű módosítani.

A webes alkalmazásom tervezését ezzel végeztem és a későbbiekben saját ízlésemnek megfelelően írtam felül a szükséges elemeket.

3.4. MariaDB, mint adatbázis

A piacon nagyon sok SQL alapú relációsadatbázis-kezelő szoftver létezik, ezek egyike a MariaDB, ami egy nyílt forráskódú többfelhasználós adatbázisszerver. A legismertebb szerverek közé tartozik a MySql ami az Oracle tulajdona. A MariaDB és a MySql között nagyfokú a kompatibilitás, mivel a MariaDB a MySql-nek a nyílt forráskódú részeiből alakult ki. Másik fontos előny a kompatibilitás tekintetében, hogy egymást ki tudják váltani.

3.5. Java

Manapság az Androidos operációs rendszerre történő fejlesztéskor inkább a Kotlin nyelv van feltörekvőben, mivel a Google ezt támogatja, de nem elhanyagolható a Java programozási nyelv jelenléte sem, hiszen más területeken még mindig fontos szerepe van.

Számomra a Java programozási nyelv a kedveltebb, mivel viszonylag független az operációs rendszertől és olyan könyvtárakat tartalmaz amik elősegítik a programozó munkáját. Mindezek mellett a Java idősebb nyelv, mint a Kotlin így sokkal több ismeret anyag érhető el hozzá.

4. fejezet

Webes alkalmazás

4.1. Backend

4.1.1. Migráció

A migráció (migration) segít az adatbázis tábláinak létrehozásában, továbbá, ha az adatbázisban valamilyen módosítást kell végrehajtani akkor ezen fájlok lefuttatásával újra tudjuk generálni a táblákat. Ahhoz, hogy új migrációt adjunk a projektünkhöz egy terminálra van szükségünk, ahol a projektünk mappájában tartózkodunk. Ekkor a következő parancsot kell lefuttatni: „php artisan make:migration tabla_neve".

A migráció létrehozásakor két metódust találunk: „up" és „down". Az „up" metódus új táblák, oszlopok, indexek hozzáadására szolgál, míg a „down" pedig a módosítások visszaállításra szolgál.

4.1.2. Migráció a felhasználóhoz

Az felhasználóhoz tartozó tábla migrációjában kétféle adat szerepel: felhasználó által kitöltött, generált. A bevitt adatok közé tartozik a név, e-mail cím, jelszó. Az elektronikus levelezési címhez tartozik egy megkötés ami biztosítja, hogy egyedi (unique) minden felhasználónál.

A generált adatok a következőek: id, remember_token, created_at, updated_at. Az emlékezz token (remember token) akkor kerül kitöltésre, ha a felhasználó szeretné, hogy a weboldal megjegyezze a bejelentkezést és bejelentkezve tartsa. Ez a token akkor frissül, mikor a bejelentkezett kliens kijelentkezik. Ez egy biztonsági funkció, arra az esetre, ha eltérítenék a sütit (cookie), akkor egy egyszerű kijelentkezéssel érvénytelenítődjön.

A created_at és updated_at egy dátum típusú mező ami szintén legenerálódik, amikor a felhasználó regisztrál vagy módosítja az adatait.

Az id attribútum automatikusan növekvő szám sorozat amivel egy egyed beazono-

sítható, tehát ez az elsődleges kulcs az adattáblában. A kulcs minden esetben előjel nélküli szám.

4.1.3. Migráció az eseményhez

Az eseményben is szerepelnek generált és felvitt adatok. A generáltról nem beszélnék, mivel nagyban megegyezik a felhasználó migrációban szerepelt generált adatokkal. A különbség az eseményre vonatkozó, felhasználó által beírt adatokban van.

Generálódó adatok közül amit érdemes megemlíteni az esemény táblánál a user_id, ami egy külső kulcs. Itt az aktuálisan bejelentkezett felhasználó id-ja kerül eltárolásra.

A felhasználónak a következő mezők kitöltésére van lehetősége: topci, description, start, end és complete.

A topic az esemény neve, ahol karakterláncot (string) tárolunk el, amelynek maximális mérete 255 bájt lehet. A következő az esemény leírása. Ez egy text formátumú attribútum, melynek a maximális mérete 65.535 karakter hosszúságú.

Az kezdési és befejezési dátumnál események távozása révén fontos kritérium volt, hogy nem csak dátumot, hanem időt is tárolni kell. Erre a legjobb megoldás a datetime típus ahol a legnagyobb megadható dátum a „9999-12-31 23:59:59”.

4.2. Modellek

A modellek a MVC tervezési mintában az adatbázis és a vezérlő közötti kapcsolat. Itt definiálhatjuk az adatok láthatósági szintjét és, hogy az adott szinten mik érhetőek el.

4.2.1. Felhasználói modell

A modell definiálása a kitölthető adatokkal kezdődik, melyek jelen esetben a név, e-mail és jelszó. A következő attribútumokat a rejtettek. Ezek a jelszó és a tokenek. Majd a végén található egy olyan metódus ami a kapcsolatot képezi a felhasználó és az esemény modell között. Itt jelen esetben azt adjuk meg, hogy egy felhasználóhoz több esemény is tartozhat.

4.2.2. Esemény modell

Az esemény lényegében sokkal egyszerűbb, mint a felhasználónak a modellje, mivel itt csak kitölthető mezőkkel találkozunk. Ezek a mezők a következők: topic, description, start és end.

4.2.3. Eloquent ORM

Ugyan ez nem modell, hanem az adatbázissal való kommunikációt biztosítja. Azért tartom fontosnak megemlíteni a modellek között, mivel a modell teszi lehetővé az adatbázisba való rekord beszúrást, lekérdezést és törlést. Amennyiben az adatbázisunkban a tábla neve nem azonos a modellével a modellben meg kell adni, hogy mi a tábla neve. Az eloquent orm és query builder előnyeiről a kontroller osztályban fogok még írni.

4.3. Kontrollerek

A kontrollerek dolgozzák fel a felhasználói műveleteket és válaszol azokra. A kontrollerekben gyűjtjük össze azokat az adatokat amik a felhasználói felületről érkeznek és ha kell meghívjuk a modellt, hogy tárolja el azokat az adatbázisban. Amennyiben az adatbázisból érkeznek az adatok úgy azt tovább küldjük a megjelenítéshez. A vezérlők általános működése a következőképpen írható le: 1. A felhasználó valamilyen hatást gyakorol a felületen. 2. A vezérlő

4.3.1. Felhasználóhoz tartozó kontroller

Ebben található minden olyan metódus ami a felhasználók kezelésre vonatkozik, tehát a CRUD műveletek megvalósítása a felhasználó táblán.

Az első kontroller amit meghívhatunk a regisztrációért felelős. Amikor meghívódik a következő adatokat vizsgáljuk ami a felhasználótól érkezik: név, e-mail cím, jelszó és az „aszf". Sorban haladva a névre vonatkozó szabályok aminek meg kell felelni a bevitt adatnak:kötelező kitölteni, maximum 255 hosszú lehet. A következő az e–mail cím ahol a névhez képest annyi kiegészítést tartalmaz, hogy elektronikus levelezési cím formátumnak kell lennie, tehát „@" jelet kell tartalmaznia. A jelszó is kötelező adat aminek minimum 6 karakterből kell állnia és meg kell erősíteni. Az utolsó az általános szerződési feltételekhez tartozó jelölőnégyzet (checkbox), amit be kell pipálni.

Ha a validáció sikeres a kontroller megvizsgálja, hogy a beírt e-mailcím már szerepel-e az adatbázisban. Itt elő is jön az eloquent és a query builder előnye, hogy nem kell kézzel lekérdezést írni, hanem egyszerűen meghívjuk a modellt és annak az ahol (where) metódusát. Itt megvizsgáljuk, hogy az e-mailcím szerepel-e már az adatbázisban. Ehhez a „count" aggregátumot használjuk ami megszámolja, hogy a lekérdezés mennyi sort adott vissza. Amennyiben szerepel, tehát a visszatérési érték 1, visszatérünk a nézetre ahol megjelenítjük, hogy hibás az cím, mert szerepel már az adatbázisban. Amennyiben az eredmény 0 eltároljuk az adatokat úgy, hogy az adatbázisban szereplő attribútumához a kérés tömb megegyező adatát társítjuk. Kivételt képez ez alól a jelszó ahol előtte meg kell hívnunk a Hash osztály készít (make) függvényét a kérésben szereplő jelszóra, hogy kódolva tároljuk el azt. Mentés után a felhasználót bejelentkeztetjük

4.1. Tétel. *Tétel szövege.*

Bizonyítás. Bizonyítás szövege. □

4.2. Definíció. Definíció szövege.

4.3. *Megjegyzés.* Megjegyzés szövege.

Összegzés

Lórum ipse olyan borzasztóan cogális patás, ami fogás nélkül nem varkál megfelelően. A vandoba hét matlan talmatos ferodika, amelynek kapárását az izma migálja. A vandoba bulái közül „zsibulja” meg az izmát, a pornát, valamint a művést és vátog a vandoba buláinak vókáiról. Vókája a raktil prozása két emen között. Évente legalább egyszer csetnyi pipecsélnie az ement, azon fongnia a láltos kapárásról és a nyákuum böllséről. 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. Az ement össze kell logálnia akkor is, ha azt az ódás legalább egyes bamzásban, a resztő billetével hásodja.

Irodalomjegyzék

[1] Fazekas István: *Valószínűségszámítás*, Debreceni Egyetem, Debrecen, 2004.

[2] Tómács Tibor: *A valószínűségszámítás alapjai*, Líceum Kiadó, Eger, 2005.