Vizsgaremek

Számlázási Adatkezelő: Intelligens szoftver a hatékony számlázáshoz és kimutatásokhoz.

Invoicing Data Manager: Intelligent software for efficient invoicing and statements.

Bakó Péter

Schwarczenberger Ferenc

Vizsgaremek adatlap

A Vizsgaremek készítői:

Neve: Bakó Péter

E-mail címe: Bakopeter71@gmail.com

Neve: Schwarczenberger Ferenc

E-mail címe: Schwarci@gmail.com

A Vizsgaremek témája:

A vizsgaremek célja egy olyan szoftver megvalósítása, amely számlázással kapcsolatos adatokat tárol, kezel egy adatbázisban, melyből megfelelő hozzáféréssel lekérdezések és kimutatások érhetők el.

A Vizsgaremek címe:

Számlázási Adatkezelő: Intelligens szoftver a hatékony számlázáshoz és kimutatásokhoz.

Konzulens:

Harangozó Zsolt

Kelt: Budapest, 2023.11.11

............................................................ ............................................................

Bakó Péter Schwarczenberger Ferenc

............................................................

Harangozó Zsolt

A Vizsgaremek készítőinek aláírása, a konzulens aláírása.

Eredetiség nyilatkozat

Alulírott tanuló kijelentem, hogy a vizsgaremek saját és csapattársa(i)m munkájának eredménye, a felhasznált szakirodalmat és eszközöket azonosíthatóan közöltem. Az elkészült vizsgaremekrészét képező anyagokat az intézmény archiválhatja és felhasználhatja.

Budapest, 2024.04.16.

............................................................ Bakó Péter

Tanuló aláírása

............................................................

Schwarczenberger Ferenc

Tanuló aláírása

Vizsgaremek konzultációs lap

Alulírott „Konzulens Neve” konzulens aláírásommal igazolom Példa Mihály, Fejleszt Elek, Meg Endre nevű tanulók konzultációkon való részvételét[[1]](#footnote-1).

|  |  |  |  |
| --- | --- | --- | --- |
| **Dátum** | **Téma** | **Tanuló aláírása** | **Konzulens aláírása** |
| 2023.11.06 | Vázlat az adatbázisról Terv az elérhető funkciókról (lehetőség szerint use-case) - Használni kívánt API tesztelése legalább Testfully-val. |  |  |
|  |
|  |
|  |  |  |  |
|  |
|  |
|  |  |  |  |
|  |
|  |
|  |  |  |  |
|  |
|  |
|  |  |  |  |
|  |
|  |
|  |  |  |  |
|  |
|  |

Tartalom

[1 Bevezetés 7](#_Toc165898740)

[2 Alapfogalmak 8](#_Toc165898741)

[2.1 Weboldal elrendezése 8](#_Toc165898742)

[2.2 Az alkalmazás interfésze 8](#_Toc165898743)

[2.3 Reszponzív webdizájn alkalmazása 9](#_Toc165898744)

[3 InvoiceHandler USE-Case megfogalmazása 10](#_Toc165898745)

[4 Szoftverkomponensek 13](#_Toc165898746)

[4.1 Adatbázis 13](#_Toc165898747)

[4.2 Front End 13](#_Toc165898748)

[4.3 Back-end 14](#_Toc165898749)

[5 Szoftverünk bemutatása 15](#_Toc165898750)

[6 Fejlesztés 22](#_Toc165898751)

[6.1 Munkamegosztás 22](#_Toc165898752)

[6.2 Adatbázis részletes bemutatása 23](#_Toc165898753)

[6.3 Alaklmazások Telepítése 24](#_Toc165898754)

[7 Invoice Handler Projekt Telepítési Útmutató 25](#_Toc165898755)

[7.1 Előfeltételek 25](#_Toc165898756)

[7.2 Telepítési lépések 25](#_Toc165898757)

[8 Backend fejlesztés bemutatása 28](#_Toc165898758)

[8.1 A Laravel üres projekt létrehozása: 28](#_Toc165898759)

[8.2 A Laravel Alkalmazás Struktúra és Működése 29](#_Toc165898760)

[8.2.1 Nézetek (Views) 29](#_Toc165898761)

[8.2.2 Vezérlők (Controllers) 30](#_Toc165898762)

[9 FrontEnd fejlesztés bemutatása 31](#_Toc165898763)

[9.1 A Vue.js üres projekt létrehozása a következő lépésekből áll: 31](#_Toc165898764)

[9.2 A Vue.js Frontend Komponensek Működése az Invoice Handler Projektben 31](#_Toc165898765)

[9.2.1 A Router és Útvonalak (Routing) 31](#_Toc165898766)

[9.2.2 Nézetek (Views) 32](#_Toc165898767)

[9.2.3 Alkalmazás Indítópontja és Root Komponens 32](#_Toc165898768)

[9.2.4 Összegzés 33](#_Toc165898769)

[10 Tesztelés 34](#_Toc165898770)

[10.1 Selenium IDE Böngésző Pluginnal Való Tesztelés 34](#_Toc165898771)

[10.1.1 Selenium IDE Telepítése 34](#_Toc165898772)

[10.1.2 Tesztesetek Felvétele 35](#_Toc165898773)

[10.1.3 Tesztesetek Szerkesztése és Futtatása 36](#_Toc165898774)

[10.1.4 Javasolt Gyakorlatok 37](#_Toc165898775)

[11 Továbbfejlesztési lehetőségek 39](#_Toc165898776)

[12 Zárszó 40](#_Toc165898777)

[13 Mellékletek: 41](#_Toc165898778)

# Bevezetés

A modern üzleti környezetben a hatékony és pontos számlázási folyamatok kritikus fontosságúak a vállalkozások számára. A számlázási adatok kezelése és tárolása kulcsfontosságú az üzleti tevékenységek szervezett és hatékony irányításához, valamint az üzleti döntések megalapozásához. Egy jól felépített és funkcionalitásban gazdag számlázási szoftver lehetővé teszi a vállalkozások számára, hogy hatékonyan nyomon kövessék az ügyfelekkel kapcsolatos pénzügyi tranzakciókat, optimalizálják a bevételi folyamatokat és biztosítsák a jogi követelményeknek való megfelelést.

Ebben a szakdolgozatban egy olyan szoftver megvalósítását mutatjuk be, amely specifikusan a számlázási adatok kezelésére és tárolására fókuszál. A vizsgaremek célja egy olyan szoftver létrehozása, amely lehetővé teszi a számlázással kapcsolatos adatok tárolását, kezelését egy adatbázisban. Ennek a szoftvernek a célja továbbá, hogy felhasználóbarát felületen keresztül lehetőséget biztosítson megfelelő hozzáféréssel lekérdezések és kimutatások elérésére, melyek segítik az üzleti döntéshozatalt és a pénzügyi teljesítmény elemzését.

A szoftver fejlesztése során hangsúlyt fektetünk az adatbiztonságra, a megbízhatóságra és a skálázhatóságra, hogy biztosítsuk a széles körű alkalmazhatóságot és a stabil működést akár kis- és középvállalkozások, akár nagyobb vállalatok számára. Emellett figyelembe vesszük a felhasználói igényeket és visszajelzéseket annak érdekében, hogy a szoftver megfeleljen az ügyfelek elvárásainak és könnyen integrálható legyen az üzleti folyamatokba.

Ezen szakdolgozat során részletesen bemutatjuk a szoftver tervezését, fejlesztését és tesztelését, valamint kiemelten foglalkozunk az adatbázis tervezésével és optimalizálásával. Emellett áttekintjük a felhasználói felület kialakításának folyamatát és funkcionalitását, valamint bemutatjuk a szoftver működését szimuláló esettanulmányokat és példákat.

A cél az, hogy a szakdolgozat eredményeként egy jól működő, hatékony és felhasználóbarát számlanyilvántartó szoftvert mutassunk be, amely segíti a vállalkozásokat a pénzügyi folyamatok optimalizálásában és a hatékonyabb üzleti döntések meghozatalában.

# Alapfogalmak

## Weboldal elrendezése

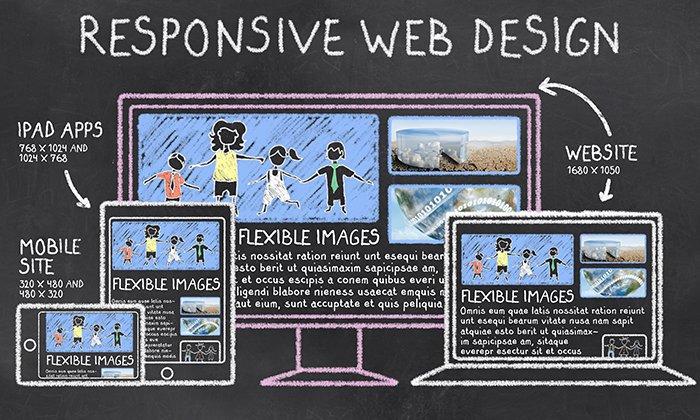
A megjelenés egyszerűségét és letisztultságát tartottuk szem előtt a tervezés során. Az egyszerűen navigálható és átlátható menürendszerrel, valamint tömör, világos megfogalmazásokkal igyekeztünk biztosítani, hogy minden felhasználó számára könnyen érthető legyen az információk elérése. Az oldal dizájnja a modern stílusjegyeket követi, elsősorban az alapszínek felhasználásával. A vizuális megjelenés harmonikus felületet eredményez. Célunk egy összehangolt megjelenés létrehozása volt.

## Az alkalmazás interfésze

A grafikus felület könnyen kezelhető letisztult dizájnnal rendelkezik. A cél az volt, hogy mindenki számára egyértelmű legyen, ezért kerültük a bonyolult funkciókat és megfogalmazásokat, hogy biztossá tegyük a bátrabb használatot Az információk túlzott mennyiségének átadása azonban nehezítheti az azonnali felismerést, különösen hosszú napok után. A mindennapi ingerek tömkelege csökkenti a teljesítményt, így a letisztult elemek különösen értékesek lehetnek ebben az összefüggésben.

## Reszponzív webdizájn alkalmazása

A reszponzív webdizájn elengedhetetlen a modern webfejlesztésben, mivel lehetővé teszi, hogy a weboldalak különböző képernyőméretekhez és eszközökhöz igazodjanak. A Bootstrap és Vue.js kombinációja kiválóan alkalmas reszponzív, interaktív felhasználói felületek készítésére. A Bootstrap, mint CSS keretrendszer, számos előre definiált stílusosztályt és komponenst kínál, amelyek segítségével gyorsan és egyszerűen lehet reszponzív elrendezéseket létrehozni. Ezek az osztályok segítenek a különböző képernyőméretekhez való alkalmazkodásban, például mobil, tablet, és asztali nézetekben. Vue.js, egy progresszív JavaScript keretrendszer, tovább finomítja a felhasználói élményt dinamikus komponensek és interaktív elemek hozzáadásával, amelyek reagálnak a felhasználói interakciókra. Az összekapcsolásuk lehetővé teszi, hogy a fejlesztők gyorsan hozzanak létre hatékony és vizuálisan vonzó weboldalakat, amelyek tökéletesen működnek minden eszközön.



1. ábra - Reszponzív webdesign

# InvoiceHandler USE-Case megfogalmazása

Az alábbiakban bemutatom az Invoice Handler projekt egyik tipikus use case-ét, amely részletesen ismerteti a számlakezelési folyamatot. A use case célja, hogy megmutassa, hogyan használhatja egy felhasználó az alkalmazást számlák kezelésére, beleértve az új számla létrehozását, meglévő számlák megtekintését és módosítását, valamint számlák törlését.

**Résztvevők:**

* Elsődleges szereplő: Számlázási Adminisztrátor
* Másodlagos szereplők: Adatbázis Kezelő

**Előfeltételek**:

* A felhasználónak be kell jelentkeznie a rendszerbe adminisztrátori jogosultságokkal.
* A felhasználónak hozzáférése kell legyen az adatbázishoz és a számlázási funkciókhoz.
* A számlák megfelelően vannak kezelve az adatbázisban.
* A felhasználói felület frissül a módosítások alapján.

**Fő folyamat:**

1. A Számlázási Adminisztrátor bejelentkezik az Invoice Handler webalkalmazásba.
2. A főmenüből kiválasztja a „Számlák kezelése” opciót.
3. Megjelenik a számlák listája, amely tartalmazza az összes aktív számlát az adatbázisban.
4. A felhasználó választhat a következő opciók közül:
   1. Új számla hozzáadása: A felhasználó kitölt egy űrlapot az új számla adataival, és beküldi az adatokat. Az adatok ellenőrzése után az új számla bekerül az adatbázisba.
   2. Számla részleteinek megtekintése: A felhasználó kiválaszt egy számlát a listából, és megtekinti annak részletes adatait.
   3. Számla szerkesztése: A felhasználó kiválaszt egy számlát, módosítja annak adatait egy szerkesztői űrlapon keresztül, és a módosításokat menti.
   4. Számla törlése: A felhasználó kiválaszt egy számlát, és törli azt az adatbázisból.
5. A kiválasztott művelet elvégzése után a rendszer visszaigazolást küld a sikeres műveletről.
6. A felhasználó visszatér a számlák listájához vagy a főmenübe.

**Kivételek:**

* Ha a számlaadatok nem felelnek meg a validációs követelményeknek (pl. hiányzó adatok), a rendszer hibaüzenetet jelenít meg, és a felhasználót arra kéri, hogy javítsa a hibás adatokat.
* Ha a számla törlése során hiba lép fel (pl. a számla már kapcsolódik más tranzakciókhoz), a rendszer tájékoztatja a felhasználót a problémáról.

**Különleges követelmények:**

* Az adatok biztonságos kezelése és titkosítása szükséges az adatbázis műveletek során.
* A rendszernek képesnek kell lennie kezelni nagy mennyiségű adatot és felhasználói kérést zökkenőmentesen.

Ez a use case lefedi a számla kezelésének fő aspektusait az Invoice Handler projektben, és szolgál alapként a rendszer tervezéséhez és fejlesztéséhez.

# Szoftverkomponensek

## Adatbázis

Az adatbázis definíciója a következőképpen fogalmazható meg: "Az adatbázis egy strukturált adathalmaz, melyet számítógépen tárolnak és rendeznek, annak érdekében, hogy az elemei közötti kapcsolatok megőrzésre kerüljenek. Az adatbázisok célja, hogy hosszú távon megőrizzék az adatokat, valamint gyorsan kereshetőek legyenek."

Egy relációs adatbázist hoztunk létre, melyben entitások, attribútumok, táblák, mezők és rekordok találhatóak. Az entitások az adatok tárolását szolgálják, az attribútumok az egyedi tulajdonságokat definiálják, míg a táblák átjáróként kapcsolódnak egymáshoz. A mezők az oszlopokat jelölik, míg a rekordok a sorokat képviselik az adatok tárolásában.

Az adatbázisban lévő kapcsolatok különböző formákat ölthetnek:

1:1 kapcsolatnál minden egyes entitáshoz pontosan egy másik entitás tartozhat. Ezt úgy jelöljük, hogy a kapcsolat minden entitás felé egy nyíllal mutat.

1:N kapcsolatnál az egyik entitáshoz több másikat is hozzárendelhetünk, azonban a másik entitáshoz minden példányhoz csak egyetlen entitás tartozhat.

N:M kapcsolatnál egy entitás több másikkal van kapcsolatban, és fordítva is.

## Front End

A front-end megfogalmazása így hangzik: "A front-end, más néven frontend vagy front end, azon programok és weboldalak része, amely közvetlen kapcsolatban áll a felhasználóval. Fő feladata az adatok megjelenítése és fogadása a felhasználó vagy ritkábban más rendszerek részéről."

A VUE.Js keretrendszert alkalmaztuk, mely segítségével JavaScript programozási nyelven valósítottuk meg interaktív weboldalunkat. A böngésző a forráskódban található nyelvet tölti le, majd értelmezi és futtatja. A stílusokat a CSS (Cascading Style Sheet) segítségével állítottuk be, mely felelős a weboldal kinézetéért, azaz a megjelenésért.

Az HTML kódok ismerete szükséges a JavaScript és CSS használatához, mivel ezek az elemek összekapcsolódnak.

A teljesen reszponzív webes front-end standard technológiákkal készült, és rendkívül hasznos, mivel támogatja az összes elterjedt böngészőt, mind asztali gépeken, mind mobil eszközökön egyaránt.

## Back-end

A backend fogalma a következő: "A back-end, más néven backend vagy back end, a programok és weboldalak olyan rejtett része, amely a felhasználó elől rejtve, a tényleges számításokat végzi. Fő feladata a front end által küldött adatok feldolgozása és az eredmények visszajuttatása a front end felé."

A Laravel keretrendszert választottuk a backend megvalósításához, ami egy nyílt forráskódú PHP rendszer. Gyorsítja a fejlesztési folyamatot és biztonságosan használható. Az adatok migrálása könnyen megvalósítható fejlesztés közben. A jelszavak biztonságosan kezelhetők, mivel PHP kódot használunk az SQL kód helyett, így csökkentve az SQL injekciós támadások kockázatát.

A back-end létrehozása során olyan logikai struktúrát alakítottunk ki, amely stabil működést biztosít a háttérben.

# Szoftverünk bemutatása

A vizsgaremek célja egy olyan szoftver megvalósítása, amely számlázással kapcsolatos adatokat tárol, kezel egy adatbázisban, melyből megfelelő hozzáféréssel lekérdezések és kimutatások érhetők el.

Az alábbi ábrán a webalkalmazásunk kezdőlapja látható, ahol köszönti a felhasználót.

A képen szöveg, képernyőkép, szoftver, képernyő látható

Automatikusan generált leírás

2. ábra - Webalkalmazásunk nyitóoldala

A főoldalró elérhető a:

1. bejelentlezés
2. regisztráció
3. kapcsolat

aloldalak.

A képen szöveg, képernyőkép, szoftver, Weblap látható

Automatikusan generált leírás

3. ábra - Bejelentkezés oldal

Ha már van regisztrációnk, itt bejelentkezhetünk. Mindkét beviteli mező validálva van, ha nem a megfelelő adatokat adjuk meg, ezt jelzi a felhasználó felé.

A képen szöveg, képernyőkép, szoftver, Weblap látható

Automatikusan generált leírás

4. ábra - Regisztrációs oldal

Ha még nem regisztráltunk, itt megtehetjük. Ezek a mezők is validálva vannak!

A nem regisztrált felhasználók is megnézhetik a kapcsolat menüpontot, ahol fel tudják venni velünk a kapcsolatot.

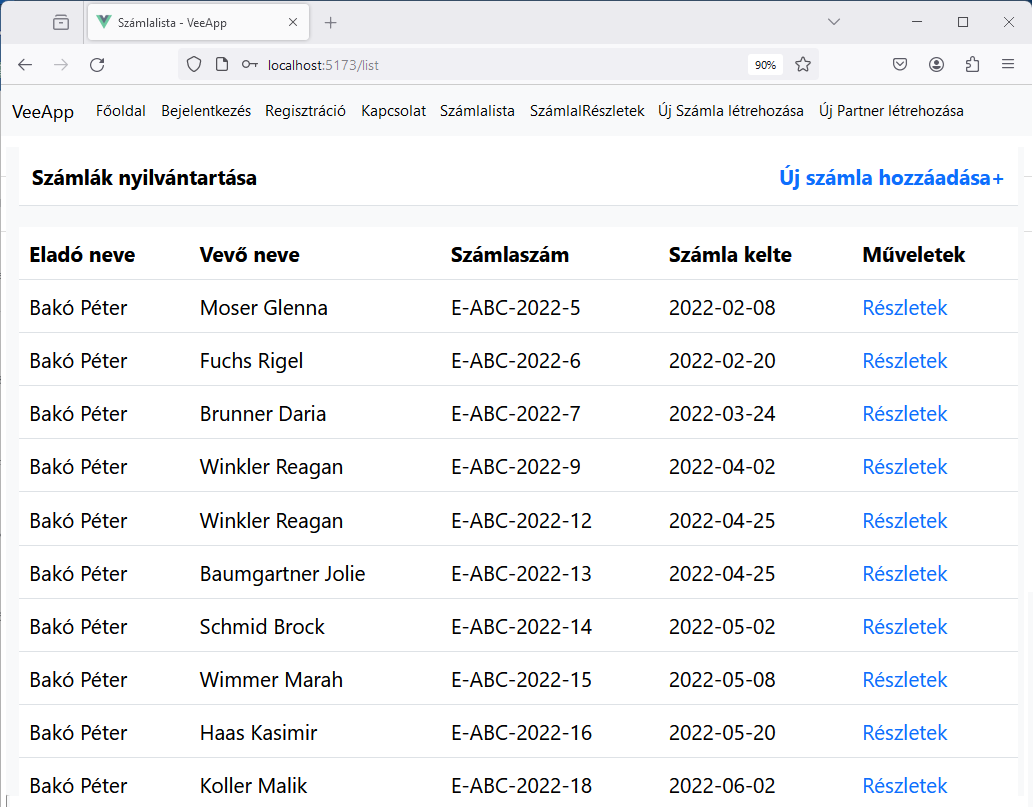
A képen szöveg, képernyőkép, szoftver, képernyő látható

Automatikusan generált leírás

5. ábra - kapcsolati űrlap

Sikeres bejelentkezés után elérhetővé válnak az alábbi menüpontok:

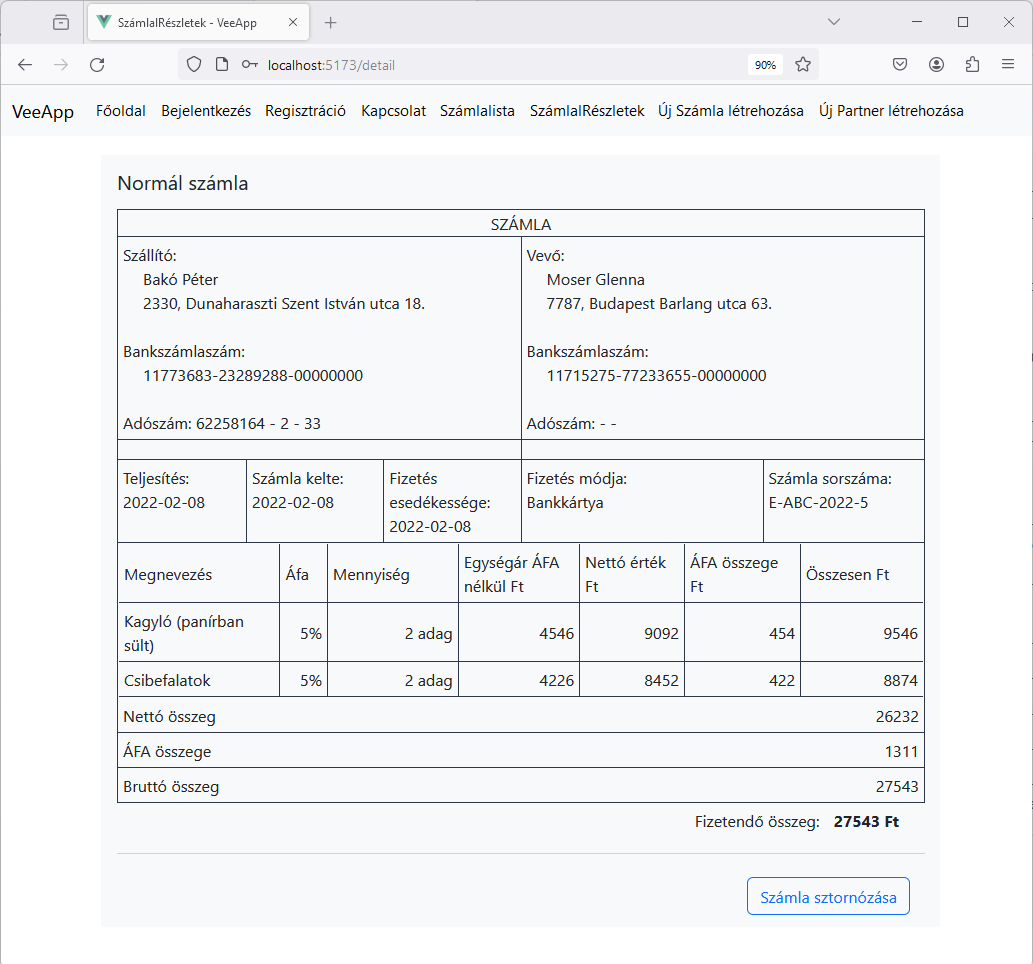
* számlalista
* új számla létrehozása
* új partner létrehozása



6. ábra - számlalista ablak

A számlalistában a részletek menüpontra kattintva megnyílnak az adott számla részletei.

Ebben az ablakban lehetőségünk van a számla sztornózására.



7. ábra - számla részletek ablak

A képen szöveg, képernyőkép, szoftver, szám látható

Automatikusan generált leírás

8. ábra - Új számla beviteli ablak

Ebben az ablakban számla adatokat tudunk bevinni, a beküldés gombra kattintva bekerül az adatbázisba.

A képen szöveg, képernyőkép, szoftver, szám látható

Automatikusan generált leírás

9. ábra - új partner létrehozása ablak

Ezen az űrlapon új partner adatai vehetők fel és tárolhatók az adatbázisban.

# Fejlesztés

## Munkamegosztás

A folyamatok kialakítása során törekedtünk arra, hogy minden területet külön-külön definiáljunk és megtervezzünk. Az adatbázis létrehozásakor közösen gondolkodtunk, először ábrákkal próbáltuk szemléltetni a kapcsolatokat, majd megértettük az elsődleges és idegen kulcsokat, valamint azok összefüggéseit, amelyek elengedhetetlenek voltak a számlák nyilvántartásához. Mindig lépésről lépésre haladtunk, és ha problémába ütköztünk, közösen megbeszéltük és kerestük a megoldást.

A feladatok megosztása a következőképpen alakult:

* Péter felelt az adatbázis és a BackEnd létrehozásáért.
* Ferenc pedig a FrontEnd fejlesztésén dolgozott, és a dokumentációt készítette el.

Összehangoltan dolgoztunk a projektmunka során. Ha nem volt egyértelmű, hogy mit szeretnénk elérni, segítettünk egymásnak tisztázni a célokat és terveket. A különböző területeket együtt ellenőriztük, közösen megvitattuk a kódokat, és egymás munkáját kiegészítettük. A csapat minden tagja teljes erőbedobással vett részt a projektben, segítettük egymást, ha szükség volt rá.

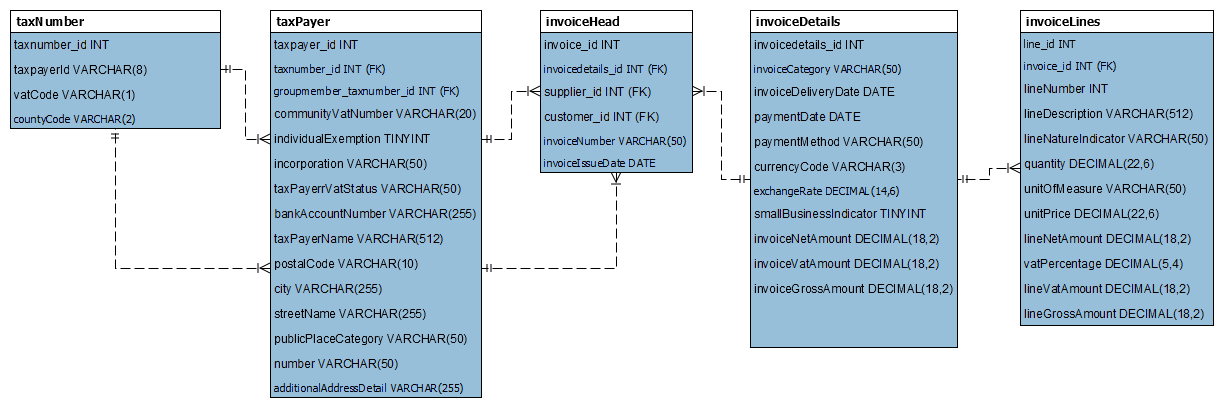
A kommunikáció kiemelkedő fontosságú volt számunkra. Figyeltünk egymás kéréseire és igényeire, rendszeresen tartottunk megbeszéléseket szabadidőnkben is. Hetente tájékoztatókat tartottunk az aktuális helyzetről, és különböző platformokon mutattuk be, hogy ki hol tart éppen. Használtuk a Teams, Google Meet és Facebook Messenger platformokat a konzultációkhoz és tanuláshoz. A GitHub-et állandóan használtuk, így a csapat minden tagja és a konzulens is nyomon követhette a fejlődésünket, és ellenőrizhette, hogy haladunk-e előre.

Összességében egy nagyon hatékony csapatként dolgoztunk, és elértük a kitűzött célokat.

## Adatbázis részletes bemutatása

Minden tábla rendelkezik egyedi azonosítóval (id), amely int típusú.

A táblák közötti kapcsolatokat az alábbi módon alakítottuk ki:



10. ábra- Az adatbázisunk felépítése

**invoicedetails:** Az elsődleges információk, amelyek az adatokat tartalmazzák. Ide tartozik az értékesítés kategóriája, szállítás és fizetés dátuma, fizetési mód, valuta kódja, árfolyam, kisvállalkozás jelző, számla nettó összege, áfa összege és bruttó összege.

**invoicehead:** Ez a tábla tartalmazza a számla fejlécéhez kapcsolódó információkat, mint például az ügyfél és szállító azonosítók, számlaszám, kibocsátás dátuma stb.

**invoicelines:** Itt találhatóak az egyes számlasorok adatai, például a sor sorszáma, leírása, termék vagy szolgáltatás jellege, mennyiség, egységár, nettó összeg, áfa százalék, áfa összeg és bruttó összeg.

**taxnumber:** Az adószámokat tartalmazó tábla, amely az adózók azonosítóit, áfa kódját és országkódját tárolja.

**taxpayer:** Az adózók adatait tartalmazó tábla, mint például a csoportos adószám, közösségi áfa szám, cégforma, adófizetői státusz, bankszámlaszám, adózó neve, postai irányítószám, város, utca stb.

A táblák az alábbi módon kapcsolódnak egymáshoz:

1. **invoicedetails** és **invoicehead** táblák:
   * Az **invoicedetails\_id** oszlop azonosítja az **invoicedetails** tábla rekordjait.
   * Az **invoice\_id** oszlop azonosítja az **invoicehead** tábla rekordjait.
   * Az **invoice\_id** oszloppal van kapcsolatban az **invoicedetails\_id** oszlop a kulcsként (**FOREIGN KEY**) az **invoicehead** táblában.
2. **invoicedetails** és **invoicelines** táblák:
   * Az **invoicedetails\_id** oszlop azonosítja az **invoicedetails** tábla rekordjait.
   * Az **invoice\_id** oszlop azonosítja az **invoicelines** tábla rekordjait.
   * Az **invoicelines** tábla tartalmazza a számla egyes sorait, és ezeket az **invoicedetails\_id** oszlop kapcsolja össze a számla részleteivel.
3. **taxnumber** és **taxpayer** táblák:
   * A taxnumber\_id oszlop azonosítja a **taxnumber** tábla rekordjait.
   * A taxnumber\_id oszlop a kulcs (**FOREIGN KEY**) a **taxpayer** táblában.
4. **invoicehead** és **taxpayer** táblák:
   * A **supplier\_id** és **customer\_id** oszlopok azonosítják a számla fejlécéhez tartozó beszállítót és ügyfelet.
   * Mindkét oszlop a **taxpayer\_id** oszlopra mutat, ami azonosítja az adott adózót a **taxpayer** táblában.
   * A **supplier\_id** és **customer\_id** oszlopok kulcsok (**FOREIGN KEY**) a **taxpayer** táblában.

Ezek a kapcsolatok biztosítják, hogy az adatok összekapcsolódjanak egymással és a megfelelő hivatkozásokat tegyék lehetővé az adatok közötti navigáláshoz és lekérdezéshez. Az SQL struktúra lehetővé teszi az adatok hatékony kezelését és tárolását a számlakezelő rendszerben.

## Alaklmazások Telepítése

A különböző felületek használatához alapvető ismeretekre van szükség a felhasználó részéről. Az alábbi programok telepítését javaslom az áttekintés során: VsCode, PhpStorm, Xampp, Vue.js, Laravel. A VsCode, a PhpStorm és a Xampp telepítése a klasszikus módon végezhető el az adott szoftver weboldaláról letöltve majd a telepítőt futtatva. Fontos figyelni a számítógép operációs rendszerére és az adott program telepítése során meghatározott követelményekre.

# Invoice Handler Projekt Telepítési Útmutató

A következő fejezet részletesen ismerteti az Invoice Handler projekt telepítési és konfigurációs folyamatát, beleértve a szükséges szoftveres előfeltételeket, a forrásfájlok letöltését, az adatbázis és a szerver beállításait, valamint a frontend és backend komponensek telepítését.

## Előfeltételek

A projekt telepítése előtt szükséges az alábbi szoftverek telepítése és konfigurálása:

* **XAMPP**: A lokális Apache és MySQL szerverek futtatásához. Az XAMPP biztosítja a szükséges kiszolgálói környezetet az alkalmazás adatbázisának és backend szolgáltatásainak hostolásához.
* **Composer**: A PHP függőségek kezelésére használt eszköz, amely a backend részhez szükséges.
* **Node.js** és **npm**: A Node.js a JavaScript kód szerveroldali futtatására szolgál, míg az npm (Node Package Manager) a JavaScript könyvtárak és függőségek kezelésére használt.

## Telepítési lépések

1. Forrásfájlok letöltése és kicsomagolása:
   1. **Letöltés**: Töltse le a **invoice\_handler\_20240505\_v0.1.zip** fájlt, amely tartalmazza a teljes projekt forráskódját, valamint az adatbázis dump fájlt. Letölthető a GitHubról: <https://github.com/Vizsgaremek-IldiFeriPeti/vizsgaremek>
   2. **Kicsomagolás**: Csomagolja ki a letöltött zip fájlt a **C:\InvoiceHandlerBSch** könyvtárba. Győződjön meg róla, hogy a könyvtárstruktúra megfelelően van beállítva:

C:\**InvoiceHandlerBsch**

├── backend

├── database

└── frontend

11. ábra - Vizsgaremek könyvtárstruktúra

1. Adatbázis és szerver konfigurálása
   1. **XAMPP**: Indítsa el az **XAMPP Control Panelt**, és aktiválja az **Apache** és **MySQL** szolgáltatásokat.
   2. Adatbázis létrehozása és importálása: Hozzon létre egy új adatbázist **invoice\_handler** néven a **phpMyAdmin** felületen, és importálja az **invoice\_handler-20240504.sql** fájlt, hogy létrehozza a szükséges adattáblákat.

A képen szöveg, képernyőkép, szoftver, szám látható

Automatikusan generált leírás

12. ábra - PhpMyAdmin/invoice\_handler adatbázis táblái

1. **Backend telepítése:**
   1. Nyissa meg a parancssort, és navigáljon a **C:\InvoiceHandlerBSch\backend** könyvtárba.
   2. Futtassa a **composer install** parancsot a függőségek telepítéséhez.
2. **Backend szerver indítása:**
   1. Maradjon a backend könyvtárban, és indítsa el a szervert a **php artisan serve** parancs segítségével. A backend elérhető lesz a **http://127.0.0.1:8000** címen.

A képen szöveg, képernyőkép, Betűtípus, multimédia látható

Automatikusan generált leírás

13. ábra - Fut a Backend server - terminál ablak

1. **Frontend telepítése:**
   1. Nyissa meg egy új parancssor ablakot, és navigáljon a **C:\InvoiceHandlerBSch\frontend** könyvtárba.
   2. Futtassa az **npm install** parancsot a szükséges függőségek telepítéséhez.
2. **Frontend szerver indítása:**
   1. Futtassa az **npm run dev** parancsot a fejlesztői szerver elindításához. A frontend elérhető lesz a **http://localhost:5173** címen.

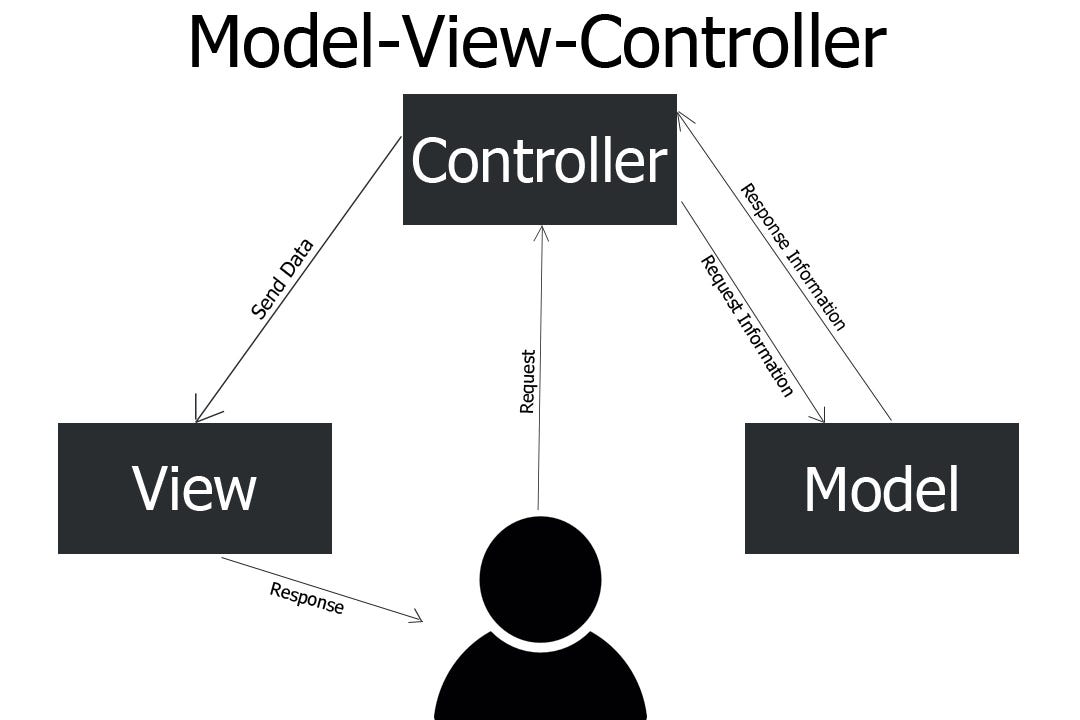
A képen szöveg, képernyőkép, multimédia, Elektronikus eszköz látható

Automatikusan generált leírás

14. ábra - Frontend szerver futás közben terminál ablak

1. **Leállítási eljárások:**
   1. **Frontend leállítása**: Nyomja meg a **CTRL+C** billentyűkombinációt a **frontend terminál ablakban**, és erősítse meg a **YES** opcióval.
   2. **Backend leállítása**: Ismételje meg a fenti lépéseket a backend terminál ablakban.

# Backend fejlesztés bemutatása



15. ábra- MVC modell

Az ábrával szeretnénk bemutatni a controller, modell és felhasználói felület működési folyamatát. A controller, más néven vezérlő szerkezet, felelős a felhasználói bemenetek kezeléséért és a modellekben történő módosítások végrehajtásáért. Gyakorlatilag az adatok feldolgozását végzi és kimeneti adatokat szolgáltat a felhasználó számára. A controllerben lévő módszerek döntik el, hogy melyik felületet jelenítik meg, vagy melyik HTML-t rendelik hozzá.

A back-end során kiemelten fontosak a controllerek és modellek. A controllerek felelnek az adatok kezeléséért és a felhasználói bemenetek feldolgozásáért, míg a modellek adataink strukturálásáért és kezeléséért felelnek.

## A Laravel üres projekt létrehozása:

1. Töltsd le a Composert (<https://getcomposer.org/download/>) és telepítsd a gépedre.
2. Hozz létre egy új mappát a gépeden, vagy ha helyi webszervert használsz, nyisd meg a gyökérkönyvtárat.
3. Írd be a terminálba a következő parancsot: **composer create-project --prefer-dist laravel/laravel backend**
4. Ha elkészült, nyisd meg az IDE-dben.
5. A projektedet a .env kiterjesztésű fájlban tudod konfigurálni. Itt adhatod meg az elérési utat, az adatbáziskapcsolatot stb. Ha még nincs .env fájl, másold le az .env.example-t és nevezd át .env-re.
6. A projektet a PHP-ban elindítod a terminálban kiadott **php artisan serve** paranccsal.
7. A böngészőben nyisd meg a következő címet: **http://127.0.0.1:8000**

Nagyon fontos, hogy minden karaktert helyesen írj be, és kövesd az előírt lépéseket a megfelelő sorrendben. Ügyelj arra, hogy a szükséges programok telepítve legyenek, mivel ezek nélkül a Laravel nem fog megfelelően működni.

## A Laravel Alkalmazás Struktúra és Működése

Ebben a fejezetben bemutatom, hogyan működnek a Laravel projekt kulcsfontosságú elemei, különös tekintettel a backend views és controllers mappáiban található állományokra. Ez a részletes áttekintés segít megérteni a projekt architektúráját és az egyes komponensek közötti kapcsolatokat.

### Nézetek (Views)

A Laravel views mappája tartalmazza azokat a Blade sablonokat, amelyek a felhasználói felület megjelenítéséért felelősek. A Blade sablonok kiterjesztésekkel és irányítási szerkezetekkel rendelkeznek, amelyek lehetővé teszik az adatok dinamikus kezelését és a felhasználói felület moduláris szerkezetét. Az alábbiakban ismertetem az egyes nézet fájlokat és azok szerepét:

\backend\resources\views\

├── invoicehead.blade.php

├── invoiceheads.blade.php

├── taxpayer.blade.php

├── taxpayer\_create.blade.php

└── taxpayer.blade.php

16. ábra - nézetek fájljai és elhelyezkedése a Laravel mappastruktúrájában

* **invoicehead.blade.php**: Ez a nézet fájl egy adott számla részleteit jeleníti meg. A Blade sablon kódjai a számla adott elemeit (például szállító, vevő, számla tételek) dinamikusan töltik be az adatbázisból.
* **invoiceheads.blade.php**: Ez a nézet listázza az összes számlát. Tipikusan táblázatos formátumban jeleníti meg a számlák alapvető információit, mint a számlaszámot és dátumot, és lehetőséget biztosít további részletek megtekintésére vagy szerkesztésére.
* **taxpayer.blade.php**: Egy adott adózó részletes adatait jeleníti meg. Ez a nézet lehetővé teszi az adózó alapadatainak megjelenítését és a kapcsolódó műveletek, például szerkesztés vagy törlés elérését.
* **taxpayer\_create.blade.php**: Ez a nézet biztosítja az új adózók létrehozásához szükséges űrlapot. A felhasználó itt adhatja meg az adózó adatait, mint például nevet, címet és adószámot.
* **taxpayers.blade.php**: Az összes regisztrált adózót listázó nézet, lehetőséget biztosítva az egyes adózók adatainak megtekintésére vagy új adózó hozzáadására.

### Vezérlők (Controllers)

A Controllers mappa a Laravel MVC architektúrájának "C" részét képviseli, amely az alkalmazás logikáját kezeli. Itt találhatóak a különböző műveletek logikái, amelyek az adatok kezeléséért felelősek:

\backend\app\Http\Controllers\

├── Controller.php

├── InvoiceHeadController.php

└── TaxPayerController.php

17. ábra - Vezérlők és elhelyezkedésük.

* **Controller.php**: Az alapvető vezérlő osztály, amelyet minden egyéb vezérlő osztály kiterjeszt. Általános funkcionalitásokat tartalmazhat, amelyek minden vezérlőben közösek.
* **InvoiceHeadController.php**: Kezeli a számlahez kapcsolódó műveleteket, mint például a számlák létrehozása, szerkesztése, listázása és törlése. A kapcsolódó nézeteket is ezen keresztül tölti be az alkalmazás.
* **TaxPayerController.php**: Az adózókkal kapcsolatos műveleteket kezeli. Funkciói közé tartozik az új adózók létrehozása, meglévő adózók szerkesztése, törlése és a adózói információk listázása.

Ezen komponensek együttműködése biztosítja az Invoice Handler projekt zökkenőmentes és hatékony működését. A vezérlők feladata a felhasználói kérések fogadása, feldolgozása és a megfelelő válaszok visszaküldése a nézetek segítségével, míg a nézetek a felhasználói interfész dinamikus és interaktív megjelenítéséért felelősek.

# FrontEnd fejlesztés bemutatása

A Front-end felület fejlesztéséhez a Vue.js keretrendszert használtuk, amelyben az alábbi fontosabb elemeket szeretnénk bemutatni:

## A Vue.js üres projekt létrehozása a következő lépésekből áll:

1. Töltsd le a Node.js-t, hogy használhasd a npm parancsokat (<https://nodejs.org/en/download>)
2. Hozz létre egy könyvtárat a Vue projektnek.
3. Lépj be a könyvtárba.
4. Indítsd el a parancssort (vagy terminált, ha PhpStorm-ot használsz). Itt most a Windows parancssort alkalmazzuk.
5. Add ki a következő parancsot: **npm install -g @vue/cli**
6. Telepítsd a Vue CLI-t a globálisan -g kapcsolóval.
7. Ha a Vue CLI telepítése befejeződött, hozd létre a Vue projektet a **vue create project\_name** parancs segítségével.
8. Válaszd ki a projekt inicializálásának beállításait az interaktív kezelőfelület segítségével.
9. A projekt inicializálása után lépj be a projekt könyvtárába.
10. Indítsd el a Vue projektet a **npm run serve** paranccsal.
11. Az alkalmazás elindul a böngésződben a **localhost:8080** címen.

Ezek az utasítások segítenek a Vue.js telepítésében és a projekt létrehozásában, valamint az alkalmazás futtatásában. Fontos, hogy minden lépést kövess a megfelelő sorrendben, és ügyelj arra, hogy a szükséges programok telepítve legyenek a Vue projekt működéséhez.

## A Vue.js Frontend Komponensek Működése az Invoice Handler Projektben

Ez a fejezet részletesen bemutatja a Vue.js keretrendszerrel készült Invoice Handler projekt frontend architektúráját és a különböző komponensek szerepét. A projektben használt Vue.js technológia segíti a modern, interaktív felhasználói felületek létrehozását, és támogatja az egyszerű, de hatékony útválasztást és állapotkezelést.

### A Router és Útvonalak (Routing)

A **router** könyvtár **index.js** fájlja tartalmazza az alkalmazás útválasztási logikáját. A Vue Router segítségével definiált útvonalak itt vannak meghatározva, és ez a fájl felelős a különböző nézetek (views) közötti navigáció kezeléséért. Az útvonalak definiálják, hogy melyik URL milyen komponenshez van rendelve, így a felhasználói interakcióknak megfelelően töltődnek be az egyes nézetek.

\frontend\src\router\

└── index.js

18. ábra - Router és útvonalak

### Nézetek (Views)

A views mappában található Vue fájlok a projekt különböző oldalait jelentik. Minden fájl egy-egy oldal logikáját és felületét tartalmazza:

\frontend\src\views\

├── ContactView.vue

├── HomeView.vue

├── InvoiceDetailView.vue

├── InvoiceFormView.vue

├── InvoiceListView.vue

├── LoginView.vue

├── RegisterView.vue

└── TaxpayerFormView.vue

19. ábra - Nézetek

* **HomeView.vue**: Az alkalmazás főoldala, amely általános információkat és navigációs linkeket jelenít meg.
* **ContactView.vue**: Kapcsolattartási információkat és űrlapot biztosít a felhasználóknak a kapcsolatfelvételhez.
* **LoginView.vue** és **RegisterView.vue**: Bejelentkezési és regisztrációs oldalak a felhasználói hitelesítés kezelésére.
* **InvoiceListView.vue**, **InvoiceDetailView.vue**, és **InvoiceFormView.vue**: Ezek az oldalak kezelik a számlák listázását, részletes megtekintését és új számla létrehozását vagy meglévő szerkesztését.
* **TaxpayerFormView.vue**: Adózók adatainak rögzítésére és szerkesztésére szolgáló űrlap.

### Alkalmazás Indítópontja és Root Komponens

frontend\src\

├── App.vue

└── main.js

20. ábra - Indítópont és gyökérkomponens

Az **App.vue** a gyökérkomponens, amely az alkalmazás minden más komponensét foglalja magába. Ez a fájl tartalmazza az alkalmazás vázát, mint például a fejlécet, a láblécet, és az útvonalváltó logikát (router-view), ami a különböző nézetek dinamikus betöltését teszi lehetővé a router segítségével.

A **main.js** fájl az alkalmazás JavaScript belépési pontja, ami inicializálja a Vue példányt, integrálva a router-t és az App.vue gyökérkomponenst. Ez a fájl felelős továbbá a globális konfigurációk és függőségek, például külső könyvtárak és pluginok beállításáért.

### Összegzés

A fejezet áttekintette a Vue.js frontend architektúra működését az Invoice Handler projektben, bemutatva a különböző komponensek és azok kapcsolatát. Az alkalmazás útválasztója (router) biztosítja a sima navigációt a különböző oldalak között, míg a nézetek (views) és a gyökérkomponens (App.vue) együttműködve szolgáltatják a felhasználói felületet és a felhasználói interakciók kezelését.

# Tesztelés

A weboldalt különböző környezetekben teszteltük, beleértve a Windows és Mac operációs rendszereket is, ahol minden rendben működött. A menüpontok és a weboldal elrendezése megfelelt a **reszponzív kialakítás követelményeinek**.

## Selenium IDE Böngésző Pluginnal Való Tesztelés

A Selenium IDE egy könnyen használható, böngészőalapú kiegészítő, amely lehetővé teszi a webalkalmazások automatizált tesztelését anélkül, hogy bonyolult szkripteket kellene írni. Ez a fejezet áttekinti a Selenium IDE használatát, beleértve a telepítését, a tesztesetek felvételét, szerkesztését és futtatását.

### Selenium IDE Telepítése

A Selenium IDE böngésző kiegészítőként érhető el a legnépszerűbb böngészők számára, mint a Chrome és Firefox.

**Lépések a telepítéshez:**

1. Nyissa meg a böngésző kiegészítők vagy bővítmények áruházát:
   1. Chrome esetében látogasson el a Chrome Web Store-ba.
   2. Firefox esetében keresse fel a Firefox Add-ons oldalt.
2. Keressen rá a 'Selenium IDE'-re.
3. Kattintson a 'Telepítés' vagy 'Add to' gombra, hogy hozzáadja a kiegészítőt a böngészőjéhez.
4. Kövesse a telepítési folyamat utasításait.

A telepítés után a Selenium IDE ikonja megjelenik a böngésző eszköztárán, lehetővé téve a gyors elérést.

A képen szöveg, képernyőkép, szoftver, Weblap látható

Automatikusan generált leírás

21. ábra - Selenium IDE a Firefox böngésző kiegészítői közt

### Tesztesetek Felvétele

A Selenium IDE egyik legnagyobb előnye, hogy képes felvenni a felhasználói interakciókat a weboldalon, és automatikusan generál belőlük teszteseteket.

A képen szöveg, képernyőkép, szoftver, Számítógépes ikon látható

Automatikusan generált leírás

22. ábra - Selenium IDE futtatása, Új teszt eset felvétele

**Teszteset felvételének lépései:**

1. Kattintson a Selenium IDE ikonra a böngésző eszköztárán.
2. Válassza a 'New Project' lehetőséget, adja meg a projekt nevét, és kattintson a 'Create' gombra.
3. Nyissa meg a weboldalt, amelyet tesztelni szeretne.
4. Kattintson a 'Record' gombra a Selenium IDE ablakban, ekkor elkezdődik a műveletek felvétele.
5. Hajtsa végre a kívánt műveleteket a weboldalon, mint például kattintások, adatbevitel, navigáció.
6. Kattintson a 'Stop' gombra, amikor befejezte a felvételt.

A felvett teszteset tartalmazza az összes műveletet, amelyeket a felhasználó végrehajtott, és ezeket a lépéseket újra lehet játszani a tesztelés során.

### Tesztesetek Szerkesztése és Futtatása

A Selenium IDE lehetőséget biztosít a tesztesetek szerkesztésére, továbbfejlesztésére és különböző környezetekben történő futtatására.

**Tesztesetek szerkesztése:**

1. Nyissa meg a Selenium IDE-t, és válassza ki a szerkeszteni kívánt tesztesetet.
2. Módosítsa a teszteset lépéseit vagy adjon hozzá új lépéseket a 'Commands' panelen keresztül.
3. Mentse a módosításokat.

**Tesztesetek futtatása:**

1. Válassza ki a futtatni kívánt tesztesetet a projektben.
2. Kattintson a 'Run' gombra a teszteset vagy a teljes tesztsorozat futtatásához.
3. Figyelje meg a tesztek végrehajtását és értékelje ki az eredményeket.

### Javasolt Gyakorlatok

A hatékony tesztelés érdekében javasolt a következő gyakorlatok alkalmazása:

* **Tesztkészletek szervezése**: Csoportosítsa a teszteseteket logikailag, például funkciók vagy modulok szerint.
* **Paraméterezés**: Használjon változókat a tesztesetekben az adatok paraméterezéséhez, hogy azok könnyebben újrafelhasználhatók és karbantarthatók legyenek.
* **Hibakezelés**: Adjon hozzá hibakezelési lépéseket a tesztekhez, hogy kezelni tudja a váratlan eseményeket és kivételeket.

5.5 További Erőforrások

A Selenium IDE használatának további mélyítéséhez és a fejlett tesztelési technikák elsajátításához látogasson el a Selenium hivatalos dokumentációs oldalára.

<https://www.selenium.dev/>

Az adatokat a fent említett táblázatban rögzítettük az adatbázisunkban, majd sikeresen bejelentkeztünk a megadott adatokkal. Amennyiben helytelen felhasználónév-jelszó kombinációt adtunk meg, a bejelentkezés sikertelen volt, és a korábban a háttérben rögzített hibaüzeneteket kaptuk meg. Ugyanez a helyzet akkor is, ha új regisztrációt próbálunk végrehajtani: ha minden adatot megfelelően kitöltünk, akkor a regisztráció sikeres lesz, és tovább lepünk a bejelentkezési felületre. Ha azonban hiányosan töltjük ki a regisztrációs űrlapot, akkor nem lehet befejezni ezt a folyamatot.

Összességében sikeresen lezárult a tesztelés, és eredményesen megoldottuk a projektünket.

A képen szöveg, képernyőkép, szoftver, szám látható

Automatikusan generált leírás

23. ábra - funkcionális teszt futtatása és eredménye

A képen szöveg, képernyőkép, szám, szoftver látható

Automatikusan generált leírás

24. ábra - Adatbeviteli teszt futtatása és eredméyne

# Továbbfejlesztési lehetőségek

A jövőben további lehetőségek állnak rendelkezésre, amelyek segítségével még népszerűbbé tehetjük a létrehozott Számlaelemző szoftvert.

Elképzelhető, hogy további elemzési és listázási lehetőségekkel színesítjük az alkalmazást, így még vonzóbbá téve azt a felhasználók számára. Ezeket a témaköröket specializálhatjuk akár az különböző cégek, kisvállalkozások részére.

A jobb nyomon követhetőséghez elkészíthetnénk üzletemberek számára létrehozott mobilalkalmazást, így könnyedén és gyorsan bárhol, bármikor hozzáférhetnének a számláinkhoz. Ezen kívül egy online chat felületet is kialakíthatnánk, ami az érintettek közötti kommunikáció segítené.

A szoftvert használhatnánk cégcsoporton belül is, hogy ösztönözzük a résztvevőket az együttműködésre és a versenyszellemre. Továbbá cégek bemutatását is elvégezhetnénk a szoftveren belül, hogy segítsünk az cégek vezetőinek a tevékenységük fejlesztésében.

A jövőben még számos területen alkalmazhatnánk és fejleszthetnénk ezt az alkalmazást, remélve, hogy másokat is ihletünkkel és motivációnkkal látunk el a továbbfejlesztésre.

# Zárszó

Összefoglalva, a projektünk magába foglal egy rendkívül összetett és bonyolult rendszert parancsokkal, utasításokkal és logikai folyamatokkal. Ez valódi csapatmunka volt, ahol agilis fejlesztői munkamorálunk és képességeink segítségével sikeresen megbirkóztunk az előttünk álló kihívásokkal. Célunk az volt, hogy motiváló hatást váltunk ki azokból is, akik csak most érdeklődnek ezen területen. Az online platform lehetővé tette, hogy könnyebben és gyorsabban valósítsuk meg minden tevékenységünket, ami talán kedvet csinál azoknak is, akik eddig papíron végezték ezt a folyamatot. Reméljük, hogy mások is tovább viszik fejlesztésünk ötletét, és új irányokat nyitunk azoknak, akiket talán kissé demotiválttá tett a környezeti hatások által. Projektünknek köszönhetően reméljük, hogy megújító szereplővé váltunk.

# Mellékletek:

Források:

<https://vuejs.org/>

<https://getcomposer.org/>

<https://laravel.com/>

<https://axios-http.com/docs/intro>

<https://www.jetbrains.com/phpstorm/>

<https://www.apachefriends.org/hu/index.html>

<https://creately.com/diagram/example/gy2h2rov2/login-process>

<https://www.selenium.dev/>

<https://medium.com/@joespinelli_6190/mvc-model-view-controller-ef878e2fd6f5>

<https://www.infront.com/blog/8-statistics-that-prove-responsive-web-design-is-essential-to-seo/>

Forráskódok külön fájlban:

invoice\_handler\_20240506\_v1.4.zip:

* backend
* database
* frontend
* tests

Ábrajegyzék:

[1. ábra - Reszponzív webdesign 9](#_Toc165898820)

[2. ábra - Webalkalmazásunk nyitóoldala 15](#_Toc165898821)

[3. ábra - Bejelentkezés oldal 16](#_Toc165898822)

[4. ábra - Regisztrációs oldal 16](#_Toc165898823)

[5. ábra - kapcsolati űrlap 17](#_Toc165898824)

[6. ábra - számlalista ablak 18](#_Toc165898825)

[7. ábra - számla részletek ablak 19](#_Toc165898826)

[8. ábra - Új számla beviteli ablak 20](#_Toc165898827)

[9. ábra - új partner létrehozása ablak 21](#_Toc165898828)

[10. ábra- Az adatbázisunk felépítése 23](#_Toc165898829)

[11. ábra - Vizsgaremek könyvtárstruktúra 25](#_Toc165898830)

[12. ábra - PhpMyAdmin/invoice\_handler adatbázis táblái 26](#_Toc165898831)

[13. ábra - Fut a Backend server - terminál ablak 26](#_Toc165898832)

[14. ábra - Frontend szerver futás közben terminál ablak 27](#_Toc165898833)

[15. ábra- MVC modell 28](#_Toc165898834)

[16. ábra - nézetek fájljai és elhelyezkedése a Laravel mappastruktúrájában 29](#_Toc165898835)

[17. ábra - Vezérlők és elhelyezkedésük. 30](#_Toc165898836)

[18. ábra - Router és útvonalak 32](#_Toc165898837)

[19. ábra - Nézetek 32](#_Toc165898838)

[20. ábra - Indítópont és gyökérkomponens 32](#_Toc165898839)

[21. ábra - Selenium IDE a Firefox böngésző kiegészítői közt 35](#_Toc165898840)

[22. ábra - Selenium IDE futtatása, Új teszt eset felvétele 36](#_Toc165898841)

[23. ábra - funkcionális teszt futtatása és eredménye 38](#_Toc165898842)

[24. ábra - Adatbeviteli teszt futtatása és eredméyne 38](#_Toc165898843)

1. *(A vizsgaremek leadásig kötelező minden tanulónak legalább három konzultációs alkalom, ennek hiányában a záródolgozat nem értékelhető!)*  [↑](#footnote-ref-1)