

Szegedi Gazdasági Szakképző Iskola
Vasvári Pál Tagintézménye

A 506131203 számú
Szoftverfejlesztő és -tesztelő
szakképesítés záródolgozata

PROJEKTFELADAT

CommerceSync Bundle

Termékkezelő alkalmazáscsomag
WooCommerce weboldalak számára

Készítette:
Bagi Péter
Pernyész Dániel
Teszár Péter

Szeged
2024

Tartalomjegyzék

BEVEZETÉS.....	3
FEJLESZTŐI DOKUMENTÁCIÓ.....	4
Komponensek.....	4
A fejlesztői csapat tagjai és feladatköreik.....	4
FUNKCIONÁLIS KÖVETELMÉNYEK.....	5
Az asztali alkalmazás funkciói:.....	5
A WordPress plugin funkciói:.....	6
A Dokumentációs oldal.....	7
Folyamatábra a programok működéséről.....	8
További fejlesztési lehetőségek.....	9
TECHNIKAI KÖVETELMÉNYEK - TECHSTACK.....	9
Frontend.....	9
Backend.....	10
Fejlesztői környezet.....	10
Dokumentáció, Kommunikációs eszközök és Project Management a fejlesztés során.....	11
A Repository.....	11
WordPress (alap) mappastruktúra:.....	12
A „végső” megoldás.....	12
Projekt Mérföldkövek.....	13
WOOSYNC - A WORDPRESS BŐVÍTMÉNY.....	14
A PRODUCTBRIDGE ASZTALI ALKALMAZÁS.....	23
ProductBridge mappa struktúra.....	23
Adatbázis.....	25
FELHASZNÁLÓI DOKUMENTÁCIÓ.....	33
Rendszerkövetelmények.....	33
A WooSync bővítmény installálása.....	34
ProductBridge Desktop alkalmazás installálása.....	38
TESZTELÉSI DOKUMENTÁCIÓ.....	40
JEGYZÉK.....	44
FELHASZNÁLT IRODALOM.....	44

BEVEZETÉS

A CommerceSync Bundle projekt célja, hogy azon WordPress és Woocommerce felhasználók számára, akik nem rendelkeznek magasabb szintű ismeretekkel a WordPress/Woocommerce használatát illetően, egy letisztult, egyszerű felhasználói felületet kapjanak, ahonnan könnyen és egyszerűen feltölthetik a termékeiket webáruházukba.

A kiinduló pontot azok az ügyfelek adták, akik számára nehézséget jelentett a termékek feltöltése a webshop-jukba. Ez érthető, hiszen sok esetben ezek a felhasználók egyéni vállalkozók, vagy olyan kisvállalkozások, családi vállalkozások, ahol nem alapvető kompetencia az informatika ismerete. Egy kézműves termékeket alkotó családi manufaktura számára nem létkérdés az informatikai ismeretek elsajátítása, hiszen Ők a kezeik által hozzák létre a portékájukat. Esetleg kézzel állítják ki a számlát vagy nyugtát is a vevőik számára. De most szeretnének növekedni, szélesebb vevőkört kiszolgálni, éppen ezért az internet felé fordulnak és webáruházat nyitnak. Egy webshop üzemeltetése önmagában kihívás lehet bárki számára, különösen ha az adott személy, vagy vállalkozás nem rendelkezik ehhez kapcsolódó ismeretekkel. A fizetést, számlázást és egyéb feladatokat természetesen lehet automatizálni, de a termékek feltöltését - hogy a példánál maradjunk - különösen egyedi kézműves termékek esetén igazán nehézkesen lehet csak. Hogy ne a fejlesztőt másnéven a "webest" keljen minden termék feltöltésekor hívni, a termék alkotója szeretné maga feltölteni azt a webáruházba.

A WordPress és WooCommerce adminisztrációs felülete pedig számtalan útvesztőt ad a felhasználóknak. Túl sok opció és túl sok hibalehetőség, amely során a felhasználó elveszik a részletekben, ezért nem tudja hatékonyan kezelni a felületet. Nem mindegyik vállalkozás tud vagy akar ilyen jellegű skill-eket elsajátítani, esetleg nem szeretne ilyen kompetenciájú munkavállalót csak ezen feladatok ellátására alkalmazni.

Több ügyféltől is érkezett visszajelzés, hogy a WordPress adminisztrációs felülete számukra bonyolult és nehézséget jelent az, hogy a webshopjukban lévő termékeket, termékkészletet naprakészen tartsák. Elmondásuk szerint annyi a menüpont, a kattintható elem, almenüpont, elrejtett gomb és opció, hogy igazán megterhelő számukra a termékfeltöltés. Ez adta az ötletet, hogy létrehozzunk egy olyan eszközcsoportot, amely segíti a hatékonyságukat. A projekt során létrehozott programcsomaggal arra fogunk törekedni, hogy minél letisztultabb, minél felhasználóbarátabb eszközt adhassunk a korábban említett célközönség részére. Hiszen ha valamit lehet egyszerűen, akkor miért bonyolítsuk. A mai világban számtalan példa van arra, hogy a felhasználók, vagy bármilyen eszközhasználó célcsoport igényli az egyszerűséget. Gondoljunk csak

a Google keresőjére, vagy az Apple termékek letisztultságára. Így a mi célunk sem kevesebb, mint, hogy a sok opcióból alkossunk keveset. Pontosabban, hogy redukáljuk az extra gombok, extra körök számát.

A WordPress egy nyílt forráskódú keretrendszer, amely bárki számára elérhető. Éppen ezért hatalmas felhasználótáborral rendelkezik. 2023-ban az interneten található weboldalak 43.2%-a WordPress motorral fut¹. Az alapvetően blogolás célzattal létrehozott keretrendszer ma már számtalan funkciót képe ellátni, amely jócskán túlmutat a weboldal fogalmán.

A WordPress weboldalak egyik bővítménye a WooCommerce, amely webáruház funkciókkal ruházza fel a weboldalt. Az online eladások 7% WooCommerce webáruházakon keresztül történt 2023-ban².

Éppen ezért találtuk fontosnak, hogy egy olyan program csomag – Bundle készüljön a felhasználók számára, amely az alapvető adminisztrációs feladatokat tesz egyszerűvé és könnyen kezelhetővé mint a termékek feltöltése vagy módosítása.

FEJLESZTŐI DOKUMENTÁCIÓ

Komponensek

A programcsomag két komponensből áll:

1. **Desktop applikáció:** ProductBridge
2. **WordPress bővítmény:** SyncWoo

Illetve létrehozunk egy dokumentációs oldalt is, ahol igyekszünk minden segítséget és instrukciót megadni a felhasználók számára.

A fejlesztői csapat tagjai és feladatköreik

Bagi Péter (fejlesztő és tesztelő):

- WooSync – Frontend és Backend
- Product Bridge – Frontend
- Dokumentáció

Pernyész Dániel (fejlesztő és tesztelő)

¹ <https://www.manaferra.com/wordpress-statistics/>

² <https://dataprot.net/statistics/woocommerce-usage-statistics/>

- Product Bridge – Front end és Backend, Adatbázis
- WooSync – Adatbázis
- Dokumentáció

Teszárý Péter (fejlesztő és tesztelő, projekt manager)

- WooSync – Frontend és Backend, Adatbázis
- Product Bridge – Frontend és Adatbázis
- Design tervek
- Dokumentáció és dokumentációs oldal
- Project management

FUNKCIONÁLIS KÖVETELMÉNYEK

Az asztali alkalmazás funkciói:

Az asztali alkalmazás segítségével a felhasználók termékeket tölthetnek fel a webshopba, illetve a felhasználók módosítására is lehetőség nyílik. Egyszerű termékek adatait fogják tudni rögzíteni. Az alkalmazás egy letisztult felülettel rendelkező, felhasználóbarát kezelést biztosító minél egyszerűbb alkalmazás létrehozása a cél. Kevés menüpont és kevés "rontási" lehetőség. A termékeket fel lehet tölteni a következő adatok segítségével:

- Termék neve
- Termék ára
- Akciós ár
- Kategória
- Termékkészlet
- SKU (Stock keeping unit)
- Tagek
- Rövid leírás
- Hosszú leírás
- Termék borítókép
- Termék galéria

Továbbá lehetőségük van a webáruházban szereplő termékek módosítására és törlésére is. Ehhez kapcsolódóan lehetőség lesz a termékek keresésére ID, név, kategória vagy SKU alapján. A felhasználókezelés kapcsán szintén lehetőség lesz új felhasználók hozzáadására, törlésére és módosítására. A felhasználóknál a következő adatok kezelésére lesz lehetőség:

- Felhasználónév
- Email cím
- Jelszó
- Felhasználói szín (User Role)

Természetesen a felhasználók között is lesz lehetőség a keresésére, szűrésre a felhasználónév, felhasználó azonosító, felhasználó email és felhasználói szint alapján.

Egy másik menüpontban lehetőség lesz megadni a kezelendő webshop URL-jét, illetve az autentikációhoz szükséges kulcsokat. Ezek segítségével jön létre a kapcsolat a webshop és az asztali alkalmazás között. A "Reset" alkalmazásakor törlődik az URL, a kulcs és a tárolt termékek is az asztali alkalmazásból. Vagyis reseteljük a programot.

Lehetőség lesz a termékek szinkronizálására is, de a desktop felületről csak azonnali szinkronizációra lesz lehetőség.

A WordPress plugin funkciói:

A WordPress bővítmény segítségével lehet majd létrehozni az összeköttetést az asztali alkalmazás és a webshop között. A kapcsolat létrehozásához egy kulcsot vagy kulcspárt kell generálni amely az autentikációt szolgálja. Ezeket a kulcsokat lehet törölni és újakat generálni amennyiben szükséges Ezeket a kulcsokat kell majd az asztali alkalmazásban megadni az erre fejlesztett felületen. A bővítmény másik funkciójának segítségével pedig CronJob-okat, lehet majd létrehozni. Pontosabban azt határozhatjuk meg, hogy a szinkronizáció mikor történjen. Lehet az havi, heti, napi, óránkénti vagy akár percenkénti szinkronizáció is. De akár lehet azonnali is. Természetesen rendelkezik majd a bővítmény egy menüvel is, ahol a menüpontokat elérhetjük, illetve további információkhoz juthatunk a programcsomaggal kapcsolatban. Lehetőség lesz elérni a dokumentációs oldalt, illetve letölteni az asztali applikációt. Az információs menüpontban pedig néhány fontos információ is helyet kapna, például, hogy a CronJob-ok használatát körültekintően érdemes alkalmazni, mert ha túl rövid a szinkronizációk közötti idő intervallum, az a szerver túlterhelését és a weboldal átmeneti összeomlását is jelentheti. Hiszen ha egyszerre túl sok kérésé

érkezik, akkor az megterhelheti a szerveret. A bővítmény fejlesztése során is szeretnénk szem előtt tartani a letisztultság és egyszerűség elvét.

A Dokumentációs oldal

A dokumentációs oldal az alkalmazáscsomag felhasználóinak informálásának céljából jön létre. Tehát egyszerűen kezelhető, letisztult felületre van szükség. Elég ha minden információ könnyen elérhető, az elérhetőségek is megtalálhatóak, illetve maga a programcsomag is letölthető az oldalon keresztül a Github repositoryval együtt. Tehát itt szükség lesz egy kezdőlapra, illetve egy dokumentációs oldalszerkezetet lehetővé tevő keretrendszerre. A dokumentációs oldal VitePress segítségével fog megvalósulni.

A VitePress egy modern, statikus weboldal-generátor, amelyet elsősorban a Vue.js keretrendszerben írt weboldalak és dokumentációk gyors és hatékony létrehozására terveztek. A VitePress lehetővé teszi a fejlesztők számára, hogy könnyen és gyorsan hozzanak létre szép és gyors weboldalakat, különösen akkor, ha a Vue.js-t használják.

Néhány kulcsfontosságú jellemző:

Vue alapú: A VitePress Vue.js-re épül, így azok számára ideális választás, akik már ismerik és szeretik a Vue.js-t. Ez lehetővé teszi a fejlesztők számára, hogy a Vue.js erőteljes funkcióit kihasználják a weboldalak létrehozásához és testreszabásához.³

Markdown alapú dokumentáció: A VitePress lehetővé teszi a dokumentáció írását Markdown formátumban. Ez a gyakran használt szövegszerkesztő formátum egyszerű és könnyen érthető módot kínál a dokumentumok írására, miközben lehetőséget biztosít a formázásra és strukturálásra.⁴

Gyors és hatékony: A VitePress a Vite alapú környezetet használja a gyors és hatékony fejlesztés érdekében. A Vite egy modern fejlesztői eszköz, amely rendkívül gyors építési és fejlesztési időt biztosít a Vue alkalmazások számára.

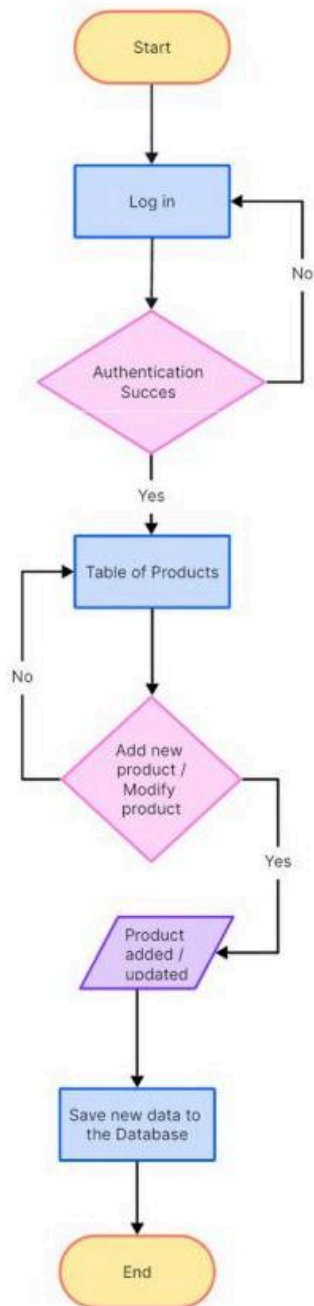
Testreszabhatóság: A VitePress testre szabható és bővíthető, lehetővé téve a fejlesztők számára, hogy saját igényeiknek megfelelően alakítsák és formázzák a weboldalakat és dokumentációkat.

³ <https://vitepress.dev/guide/what-is-vitepress>

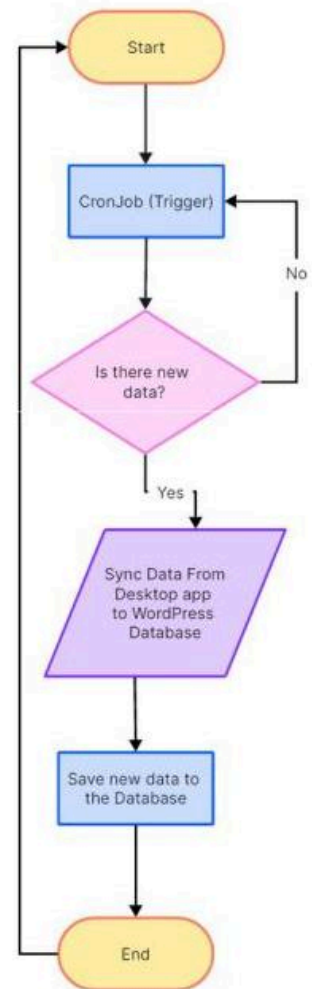
⁴ <https://www.markdownguide.org/getting-started/>

Folyamatábra a programok működéséről

ProductBridge (Desktop App)



WooSync WordPress Plugin



1. ábra: Folyamatábra a programok működéséről

További fejlesztési lehetőségek

Az jelenleg megvalósítandó funkciók mellett további lehetőségek, fejlesztési opciók is számításba kerültek. Ezekről röviden:

- **Készletkezelés** - Jelenlegi terveinkben csak az szerepel, hogy a feltöltés során megadható legyen, hogy egy adott termékből hány darab áll rendelkezésre. Viszon új funkcióként komplex készletkezelést is lehetővé lehetne tenni.
- **Variálható termékek feltöltése** - Az első kiadás csak egyszerű termékek feltöltésére biztosítana lehetőséget. A jövőben lehetőség nyílna variálható termékek feltöltésére is. Ennek nehezítő körülményei a szabadon létrehozható termékattribútumok.
- **Tömeges termékmódosítás** - A termékek tömeges módosítása lehetővé tenné például a százalékos leértékelések megvalósítását nagy tételben.
- **Multisite webshopok kezelése** - A WordPress és ez által a WooCommerce lehetőséget biztosít úgynevezett Multisite-ok létrehozására is. Ez azt jelenti, hogy egy WordPress installáláson belül több WordPress oldal is lehetséges. Vagyis egy felületen bejelentkezve, több webshopot is kezelhetünk.⁵ Tehát ennek a jövőbeni fejlesztésnek a kihívása, hogy ilyen típusú Multisite-ok kezelését is lehetővé tegye a programcsomag.
- **Több webshop egyidejű kezelése** - Ez nem összetévesztendő a multisite-al. Itt több URL és több kulcs hozzáadására volna lehetőség az asztali applikációban, vagyis egyszerre, egy felületről több különböző webshopot is lehetne kezelni.

TECHNIKAI KÖVETELMÉNYEK - TECHSTACK

Frontend

- **HTML:** A WordPress bővítmény fejlesztéséhez a HTML leírónyelvet használtunk, amely a felület elrendezéséért és részben a tartalom megjelenítéséért felel.
- **CSS:** Bizonyos elemekhez saját CSS-t használunk, hogy az arculathoz illő színeket, elrendezéseket megvalósítsuk.
- **Bootstrap:** A továbbfejleszthetőség és konzisztencia miatt a Bootstrap CSS keretrendszert alkalmazzuk. A felület kialakítását a keretrendszer osztályainak használatával valósítjuk meg, így biztosan minden felületen megfelelően fog működni a bővítmény.

⁵ <https://developer.wordpress.org/advanced-administration/multisite/create-network/>

- **JavaScript**et és **jQuery**-t használunk bizonyos logikai feladatok ellátására, illetve az API kulcsok generálásában is segíti a bővítményt.
- **Windows Presentation Foundation** (WPF) keretrendszer: A desktop applikáció fejlesztése során a WPF keretrendszert és C# programozási nyelvet használunk. Ezáltal lehetőségünk nyílik, hogy egy jól kezelhető, jól működő natív applikációt fejlesszünk Windows operációs rendszerre.
- **Vue.js**: A Vue.js keretrendszer segítségével építjük fel a dokumentációs oldalt.
- **Vite**: A Vue.js keretrendszer egy modern és gyors fejlesztői tool, amelyet a Vue.js-el együttesen használva alakítjuk ki a dokumentációs oldalt.
- **Markdown**: A Markdown nyelv segítségével töltjük fel az információt a dokumentációs oldalra.

Backend

- **MySQL**: A desktop applikáció mögött egy MySQL adatbázis fut. Ebben tároljuk a termékek, felhasználók adatait és az autentikációhoz szükséges kulcsokat, URL-eket.
- **MariaDB**: A WordPress oldalunkhoz MariaDB-t használunk. Mivel szükség van a WordPress adattábláinak módosítására, illetve saját táblákat is létre kell hozni, ezért a WordPress saját adatbázisához is hozzá kell nyúlnunk.
- **C#**: A backend működésének funkcióit és logikáját C# nyelven írjuk az MVVM (Model, View, ViewModel) pattern szerint.
- **PHP**: A bővítmény PHP nyelven íródik. A PHP-ba ágyazott HTML és CSS segítségével kapjuk meg a kinézetet, a JavaScript és jQuery segítségével bizonyos funkciókat, de az alapvető működésért a PHP felel a bővítmény esetében.

Fejlesztői környezet

A fejlesztés során a következő eszközöket hítuk segítségül:

- **Git**: A Git verziókezelő segítségével követjük nyomon a kódbázis változásait, a különböző brancheket és commitokat.
- **Github**: A Github szerverein tároljuk a fejlesztéssel kapcsolatos kódbázist az azok számára létrehozott repository-ban.

- **Github Pages:** A Github Pages segítségével keltjük életre és tesszük mindenki számára elérhetővé a dokumentációs oldalt. Így egy költséghatékony folyamatosan rendelkezésre álló felület szolgálja ki a felhasználókat.
- **Visual Studio Code:** A bővítmény teljes egészét a VS Code segítségével fejlesztettük. Remek eszköz és számtalan hasznos kiegészítő áll rendelkezésre. Így minden egy helyen oldható meg.
- **Visual Studio:** Ez az IDE számtalan funkcióval rendelkezik amelyek segítségével a desktop applikáció fejlesztése végbement. Segítségével egy programon belül láthatjuk és írhatjuk meg a frontend és backend részét a desktop applikációnak.
- NuGet csomagkezelő
- **XAMPP:** Az Xampp segítségével futtatjuk a szoftvereink működéséhez szükséges MySQL és MariaDB adatbázisokat, illetve a működéshez szükséges Apache szerveret.
- **Postman:** A postman segítségével terveztük, hoztuk létre és ellenőriztük az API végpontokat.
- **WordPress:** A fejlesztéshez Localhost-on hoztuk létre a tesztoldalt. A WooCommerce segítségével pedig a webáruházat, melynek termékeit kezelni fogjuk.

Dokumentáció, Kommunikációs eszközök és Project Management a fejlesztés során





A fejlesztéssel kapcsolatos dokumentáció az Atlassian szoftvercsalád Confluence nevű applikációjában történik. Itt alakítottuk ki a Wiki-t, A dokumentáció és tesztdokumentumok, illetve minden egyéb kapcsolódó dokumentum helyét is, például mint ez a fejlesztői dokumentáció. A feladatok nyomon követése és delegálása pedig szintén az Atlassian szoftvercsalád Jira nevű applikációjában történik, valamint a Github Projects segítségével. Itt jelöljük ki a milestone-okat. Osztjuk szét a feladatokat, modulonként.

A csapat Emailen és egy külön erre a célra létrehozott Space-en keresztül kommunikál a Google Chat applikáción belül. Heti rendszerességgel tartunk meetingeket, stand up-okat. Ugyan akkor a dokumentációkat feltöltjük a Github Repository-ba is, illetve a dokumentációs oldalra.

A Repository

A WordPress bővítmény sajátossága az, hogy külön mappában kap helyet a bővítmény és külön mappában kap helyet a téma. Ugyan ez vonatkozik a Desktop applikációra.

WordPress (alap) mappastruktúra:

- └─  wordpress
 - └─  wp-admin
 - └─  wp-content
 - └─  wp-includes
 - └─ .htaccess
 - └─ index.php
 - └─ license.txt
 - └─ readme.html
 - └─ wp-activate.php
 - └─ wp-blog-header.php
 - └─ wp-comments-post.php
 - └─ wp-config-sample.php
 - └─ wp-config.php
 - └─ wp-cron.php
 - └─ wp-links-opml.php
 - └─ wp-load.php
 - └─ wp-login.php
 - └─ wp-mail.php
 - └─ wp-settings.php
 - └─ wp-signup.php
 - └─ wp-trackback.php
 - └─ xmlrpc.php

Amint látjuk a pluginek és témák külön almappákban kapnak helyet. Ezért nehezítő körülmény, hogy a projekt esetében egy darab repository tartalmazza az összes fájlt, programot. Annál is inkább, mert fejlesztés közben tesztelnünk is kell mindent. Folytak kutatások azzal kapcsolatban miként oldható meg, hogy egy repository almappái külön helyeken legyenek lokálisan „lehúzva”, de nem működött egyik megoldás sem.⁷

A „végső” megoldás

Tehát végső megoldásként, illetve, hogy a munka is haladhasson, ne pedig a technikai feltételek, kritériumok vegyék el az időt a munkától, az az elhatározás született, hogy a három komponens külön repositoryban kerül fejlesztésre. Majd a fejlesztés végeztével, a kész projektet

⁶ <https://stackoverflow.com/questions/7106012/download-a-single-folder-or-directory-from-a-github-repo>

⁷ <https://www.wpbeginner.com/beginners-guide/beginners-guide-to-wordpress-file-and-directory-structure/>

feltöltjük a Bundle repositoryba. A Bundle Readme.MD fájl pedig tartalmazni fogja az eredeti repository-k kapcsolódó commitjait, az egyszerűbb ellenőrzés érdekében. Tehát végeredményben minden kritérium teljesülni fog, viszont a saját munkánkat tesszük egyszerűbbé a fejlesztés során.

Az alábbiak szerint fognak kinézni a repository-K:

Main/Bundle Repository:

CommerceSync-Hub-Bundle

<https://github.com/2023e-vp-vizsgaremek/e-commerce>

Desktop Application Repository:

Product-Bridge

(<https://github.com/CommerceSync-Hub/Product-Bridge>)

Plugin Repository:

Sync-Woo

(<https://github.com/CommerceSync-Hub/Sync-Woo>)

Dokumentációs oldal (live GitHub Pages oldal):

<https://commercesync-hub.github.io/commerce-sync-docs/>

Projekt Mérföldkövek

A Projekt tervezésekor igyekeztünk reális célokat és dátumokat kitűzni magunk elé. A hatékonyság miatt fontos volt, hogy lássuk, melyek azok a deadline-ok, amelyekhez igazodnunk kell. Ezáltal időben is el tudtuk osztani a feladatokat, és fázisokat, hogy legyen a projektünknek eleje és vége.

A tapasztalat is bizonyítja, hogy igenis fontos a tervezés. Különben semminek nincs gazdája, és a feladatok nem lesznek elvégezve. Tehát ha sikerül meghatározni az optimális és mindenki számára tartható keretet egy idő intervallumon belül, nagy valószínűséggel tartható lesz a tempó.

Minden esetre, ha látjuk a dátumokat, akkor jó meghatározott feladatokkal, megtervezett lépésekkel, és megfelelő előkészületekkel, tarthatóak a határidők.






Milestone #1	2023. október 15.	A Bundle működési szerkezetének kialakítása, fejlesztőkörnyezetek létrehozása, Project management felületek kialakítása, a fejlesztői csapat összeszervezése. Tech Stack meghatározása.
Milestone #2	2023. november 26.	UserFlow, Programtervek és Design terv elkészítése
Milestone #3	2024. január 15.	Az első működőképes prototípus (Alpha version) elkészítése a WordPress Témából , bővítményből és desktop applikációból alap adatokkal.
Milestone #4	2024 február 28.	Javított második kiadás (Beta version) teszteléshez.
Milestone #5	2024 március 28.	Javított végleges (Release candidate) kiadás.

WOOSYNC - A WORDPRESS BŐVÍTMÉNY

A WooSync bővítmény képzi a hidat a webshop (WordPress/WooCommerce) és a desktop applikáció (ProductBridge) között. Fontos, hogy ez a komponens is egyszerűen telepíthető és kezelhető legyen. Éppen ezért úgy lett létrehozva, hogy nem igényel különösebb szaktudást az installálása és beállítása.

A következő ábrán bemutatjuk a mappastruktúrát, melyből kimeljük a fontosab kódrészleteket, fájlokat:

WooSync mappa struktúra

- └─  woosync
- └─  assets
 - └─ commercesync-logo.png
- └─  css
 - └─ bootstrap.min.css
 - └─ woosync.css
- └─  includes
 - └─ woosync-admin.php
 - └─ woosync-class.php
 - └─ woosync-sync.php
- └─  js
 - └─ bootstrap.min.js
 - └─ woosync-script.js

- └─ languages
 - └─ woosync-plugin-hu_HU.mo
- └─ navigation
 - └─ woosync-menu.php
- └─ pages
 - └─ woosync-authentication.php
 - └─ woosync-info-page.php
 - └─ woosync-settings-page.php
- └─ README.md
- └─ scripts
 - └─ woosync-scripts.php
- └─ uninstall.php
- └─ woosync.php

A modularitás érdekében külön mappákba rendeztük a különböző osztályokat, fájlokat. Ahogyan az a fenti ábrán is látható, igyekeztünk az áttekinthetőség, továbbfejleszthetőség és a karbantartás miatt logikusan és kisebb elemekre bontva struktúrálni a bővítményt. A **főmappában** kapott helyet a woosync.php vagyis a main fájl, illetve az uninstall.php. Az **assets**ben a médiaelemek kapnak helyet. A **css** mappában a stíluslapok. Az **includes**-ban a fő osztály funkciói kerülnek, illetve a kapcsolódó function-ök. A **js** könyvtár a javascript fájlokat, a **languages** a nyelvi fájlokat foglalja magában, pontosabban egyelőre csak a magyart. A **navigation** is kapott egy külön mappát, mert bár csupán egy fájlról van szó, az áttekinthetőség volt a cél. A **pages** mappába kerültek az admin menüben megjelenő oldalak kódjai.

Mint minden WordPress bővítmény esetén itt is szükség van egy fő (main) PHP fájlra, ami azért felelős, hogy a WordPress keretrendszer felismerje a programot. Ennek a main fájlra, amely a mi esetünkben a woosync.php fájlnevet kapta, minden esetben tartalmaznia kell legalább a következő kódrészletet:

```
/**
 * Plugin Name: WooSync
 * Description: A plugin to synchronize products from a desktop application to WooCommerce.
 * Version: 1.0.1
 * Author: CommerceSync Team
 * Text Domain: woosync
```

```
* Domain Path: /languages
* License: GPL v2 or later
* License URI: https://www.gnu.org/licenses/gpl-2.0.html
*/
```

A WordPress bővítmény fejlécének része, melynek számos fontos szerepe van:⁸

- **Plugin Name:** Itt adjuk meg a bővítmény nevét. Ez a név jelenik meg a WordPress admin felületén a bővítmények kezelőjében és más helyeken, így könnyen azonosítható lesz a felhasználók számára.
- **Description:** Ez a leírás rövid összefoglalása annak, hogy mi a bővítmény célja vagy funkciója. Segít abban, hogy a felhasználók gyorsan megértsék, hogy mire szolgál a bővítmény, és hogy miért érdemes-e telepíteniük.
- **Version:** Itt adható meg a bővítmény verzióját. Ez fontos információ, amely lehetővé teszi a felhasználók számára, hogy követhessék a bővítmény frissítéseit és változásait.
- **Author:** Ez a bővítmény szerzőjének nevét vagy a csapat nevét tartalmazza. Ez segít azonosítani, hogy ki fejlesztette a bővítményt, és kihez lehet fordulni esetleges kérdések vagy problémák esetén.
- **Text Domain:** Ez a szövegdomain, amelyet a bővítmény fordításához használnak. Ez lehetővé teszi, hogy a WordPress fordítási rendszere megfelelően kezelje a szövegeket és a nyelvi fájlokat.
- **Domain Path:** Ez a mappa elérési útja, ahol a fordítási fájlok találhatóak. Segít meghatározni, hogy hol kell keresni a fordítási fájlokat a bővítményben.
- **License:** Ez a bővítmény licenszének típusát adja meg. Ebben az esetben a GPL v2 vagy későbbi verziók licencét választottuk, amely egy nyílt forráskódú licenz, amely lehetővé teszi a szabad felhasználást, módosítást és terjesztést.
- **License URI:** Ez a licenz URL-jét tartalmazza, ahol a felhasználók részletesen tájékozódhatnak a bővítmény licenszének feltételeiről és jogaira vonatkozóan. Ez fontos információ a felhasználók számára a bővítmény jogi státuszának megértéséhez.

Ezeket mind kommentként kell elhelyezni. Így tudja majd értelmezni a WordPress keretrendszer a bővítmény adatait. A WordPress adminisztrációs felületén pedig a következő

⁸ <https://developer.wordpress.org/plugins/plugin-basics/header-requirements/>

képpen jelenik meg:

<input type="checkbox"/> Plugin	Description	Automatic Updates
<input type="checkbox"/> Duplicate Page Settings Donate Deactivate	Duplicate Posts, Pages and Custom Posts using single click. Version 4.5.3 By mndpsingh287 View details	Enable auto-updates
<input type="checkbox"/> WooCommerce Settings Deactivate	An ecommerce toolkit that helps you sell anything. Beautifully. Version 8.8.2 By Automatic View details Docs API docs Community support	Enable auto-updates
<input type="checkbox"/> WooSync Deactivate	A plugin to synchronize products from a desktop application to WooCommerce. Version 1.0.1 By CommerceSync Team	
<input type="checkbox"/> Plugin	Description	Automatic Updates

Bulk actions ▼ Apply

3 items

A WordPress "Plugins" menüpontja

Illetve ugyan ebbe a fájlba kerültek az alábbi utasítások is:

```
if (!defined('ABSPATH')) exit;

include_once(plugin_dir_path(__FILE__) . 'includes/woosync-main.php');

include_once(plugin_dir_path(__FILE__) . 'activation/woosync-activation.php');

register_activation_hook( __FILE__, 'woosync_create_tables' );

new WooSync();
```

Az első kódsor egy biztonsági intézkedés a WordPress környezetben. Nézzük részleteiben:

1. **defined('ABSPATH')**: Ez a rész ellenőrzi, hogy a ABSPATH nevű konstans definiálva van-e vagy sem. Az ABSPATH a WordPress fő könyvtárának elérési útját jelenti, és alapvető fontosságú a WordPress rendszer megfelelő működéséhez. Ha ez a konstans definiálva van, az azt jelenti, hogy a WordPress keretrendszer betöltődött, és a kód a WordPress környezetben fut.
2. **!defined('ABSPATH')**: Az ! (nem) operátorral ellátott feltétel azt jelenti, hogy ha az ABSPATH konstans nincs definiálva, vagyis ha a WordPress fő könyvtára nem érhető el, akkor a feltétel igaz lesz.
3. **exit**: Ha a feltétel igaz, vagyis ha az ABSPATH konstans nincs definiálva, akkor a exit függvény meghívásával azonnal megszakítódik a kód végrehajtása. Ez megakadályozza, hogy a kód tovább fusson a WordPress környezet nélkül, ami biztonsági és működési problémákat okozhatna.

Összességében ez a kódsor azt jelenti, hogy a kód csak akkor fut tovább, ha a WordPress fő könyvtára elérhető és a WordPress környezetben van. Ha nem, akkor a kód futása azonnal megszakad, hogy megakadályozza a lehetséges hibákat és biztonsági réseket. Egy nagyon fontos részlete a bővítménynek.

A további kódsorokról röviden:

- **include_once(plugin_dir_path(__FILE__) . 'includes/woosync-main.php');** Ez a sor azt a fájlt hívja meg, amelyben a másodszintű, de kiemelten fontos funkciókat tartalmazza, mint például a bővítmény menüpontjainak megjelenítése a WordPress admin felületen. A `plugin_dir_path(__FILE__)` az aktuális fájl elérési útját adja vissza, és az `includes/woosync-main.php` a beillesztendő fájl relatív elérési útját jelöli. Az `include_once` függvény beolvassa és végrehajtja a megadott fájlt, de csak egyszer, még akkor is, ha többször hívják meg.
- **include_once(plugin_dir_path(__FILE__) . 'activation/woosync-activation.php');** Hasonlóan az előzőhöz, ez a sor egy másik fájl beillesztését jelenti a jelenlegi fájlba. Ez alkalmas a "woosync-activation.php" nevű fájl beolvasására, amely a bővítmény aktiválásával kapcsolatos funkciókat tartalmazza.
- **register_activation_hook(__FILE__, 'woosync_create_tables');** Ez a sor egy olyan funkciót regisztrál, amelyet akkor hív meg a WordPress, amikor a bővítmény aktiválva lesz. Ez a funkció a bővítmény adatbázis tábláinak létrehozását végzi az aktiváláskor.
- **new WooSync();** Ez a sor egy új példányt hoz létre a WooSync osztályból. Ez a bővítmény fő osztályának példányosítása, amely tartalmazza a bővítmény fő funkcióit és beállításait.

A woosync-class.php fájl:

```
class WooSync {  
    function __construct() {  
        add_action('admin_menu', array($this, 'register_menu_once'));  
        add_action('admin_init', array($this, 'initialize_settings'));  
        add_action('admin_enqueue_scripts', array($this, 'enqueue_scripts'));  
  
        include_once(plugin_dir_path(__FILE__) . 'woosync-ajax.php');
```

```
add_action('wp_ajax_generate_api_key', array($this, 'generate_api_key_callback'));
add_action('wp_ajax_reset_api_key', array($this, 'reset_api_key_callback'));
add_action('wp_ajax_sync_and_check_changes', array($this,
'sync_and_check_changes_callback'));
```

Ez a PHP kód egy osztály definíciót tartalmaz, amelyet WooSync-nek neveztünk el. Nézzük részletesen:

- **class WooSync { ... }:** Ez létrehoz egy osztályt, amelynek neve WooSync. Az osztály tartalmazza a bővítmény fő funkcióit és metódusait.
- **function __construct() { ... }:** Ez a konstruktor metódus, amelyet akkor hív meg a PHP, amikor egy új WooSync objektumot hozunk létre. A konstruktor metódusban különböző műveletek és akciók vannak definiálva, amelyek a bővítmény inicializálását és beállítását végzik.
- **add_action:** Ez a WordPress action hook-okhoz hozzáad egy műveletet vagy callback függvényt. Ezek a hook-ok meghatározzák, hogy mikor és milyen kódot hajtson végre a WordPress futás közben. Például az admin_menu hook hozzáadja a menüpontot az admin felületen, az admin_init hook inicializálja a bővítmény beállításait, az admin_enqueue_scripts hook pedig a szkriptek és stílusok betöltéséért felelős.
- **include_once:** Ez a függvény egy másik fájl beolvasását végzi a jelenlegi fájlba. Ezáltal lehetővé teszi más fájlok tartalmának felhasználását és hozzáférését a WooSync osztályban.
- **wp_enqueue_style és wp_enqueue_script:** Ezek a függvények felelősek a WordPress-en belüli stílusok és szkriptek betöltéséért. Ezáltal biztosítja, hogy a szükséges fájlok a megfelelő módon legyenek integrálva a WordPress oldalakba.
- **wp_localize_script:** Ez a függvény lehetővé teszi a szkriptek számára, hogy adatokat küldjenek a WordPress adminisztrációs felületen kívül. Ebben az esetben az ajax_url és a security adatokat küldi a woosync-script.js fájlban.
- **A render_ metódusok:** Ezek a metódusok felelősek az adminisztrációs felület oldalainak rendereléséért. Ezek a metódusok betöltik és megjelenítik az egyes oldalakhoz tartozó tartalmakat.
- **new WooSync();:** Ez létrehoz egy új WooSync objektumot, és ezzel elindítja a bővítmény inicializálását és működését.

Összességében ez a kód a bővítmény fő osztályának definícióját tartalmazza, amely felelős a bővítmény működéséért és funkcionalitásáért.

Menüpontok regisztrálása

```
function register_menu_once() {  
    static $registered = false;  
    if (!$registered) {  
        add_menu_page('WooSync', 'WooSync', 'manage_options', 'ourwoosyncplugin',  
array($this, 'render_woosync_page'), 'dashicons-smiley', 111);  
        add_submenu_page('ourwoosyncplugin', 'Infon', 'Information', 'manage_options',  
'woosync-info', array($this, 'render_info_page'));  
        add_submenu_page('ourwoosyncplugin', 'WooSync Authentication', 'Authentication',  
'manage_options', 'woosync-authentication', array($this, 'render_auth_page'));  
        add_submenu_page('ourwoosyncplugin', 'WooSync Settings', 'Options',  
'manage_options', 'woosync-settings', array($this, 'render_settings_page'));  
        $registered = true;  
    }  
}
```

A `register_menu_once()` metódus a WordPress adminisztrációs menühöz regisztrál menüpontokat. Nézzük részletesen:

- **static \$registered = false;**: Ez a sor deklarál egy statikus változót, amelyet `false` értékkel inicializálunk. Ez a változó arra szolgál, hogy nyomon kövesse, hogy már regisztráltuk-e a menüpontokat vagy sem.
- **if (!\$registered) { ... }**: Ez a blokk ellenőrzi, hogy még nem lett-e regisztrálva a menüpont. Ha a `$registered` változó értéke `false`, akkor a blokk lefut, és regisztrálja a menüpontokat.
- **add_menu_page(...)**: Ez a függvény hozzáad egy új menüpontot az adminisztrációs menühöz. A `WooSync` menüpontot a `WooSync` címkével jeleníti meg. A `manage_options` jogosultsággal rendelkező felhasználók láthatják ezt a menüpontot. Az `ourwoosyncplugin` egyedi azonosítót kap, és a `render_woosync_page()` metódus hivatkozásával kerül megjelenítésre. A `dashicons-smiley` a menüpont ikonját jelöli, míg a `111` a menüpont súlyát határozza meg.
- **add_submenu_page(...)**: Ezek a függvényhívások hozzáadnak almenüpontokat a főmenüpont alá. Minden almenüpont más oldalra mutat, és különböző címkékkel jelennek meg a menüben. A jogosultságok és az oldalak megjelenítéséhez szükséges metódusok is meg vannak adva.

- **\$registered = true;** Miután regisztráltuk a menüpontokat, az \$registered változó értékét true-ra állítjuk, hogy jelezzük, hogy már megtörtént a regisztráció. Ez a metódus biztosítja, hogy a menüpontokat csak egyszer regisztrálják, és elkerüli a felesleges ismételt regisztrációt. A menüpontok segítségével a felhasználók könnyen navigálhatnak a bővítmény különböző funkcióihoz és beállításaihoz.

Scriptek és stíluslapok betöltése/integrálása

```
function enqueue_scripts() {
    wp_enqueue_style('woosync-css', plugin_dir_url(__FILE__) . '../css/woosync.css', array(),
'1.0');
    wp_enqueue_style('bootstrap-css', plugin_dir_url(__FILE__) . '../css/bootstrap.min.css',
array(), '5.3.0');
    wp_enqueue_script('woosync-script', plugin_dir_url(__FILE__) . '../js/woosync-script.js',
array('jquery'), '1.0', true);
    wp_enqueue_script('bootstrap-js', plugin_dir_url(__FILE__) . '../js/bootstrap.min.js',
array('jquery'), '5.3.0', true);
    wp_localize_script('woosync-script', 'woosync_ajax_obj', array(
        'ajax_url' => admin_url('admin-ajax.php'),
        'security' => wp_create_nonce('woosync_ajax_nonce')
    ));
}
```

Ez a függvény különböző CSS és JavaScript scriptek és stíluslapok betöltéséért felelős.

Adminisztrációs oldalak megjelenítése

```
function render_woosync_page() {
    include_once(plugin_dir_path(__FILE__) . '../pages/woosync-info-page.php');
}
function render_info_page() {
    include_once(plugin_dir_path(__FILE__) . '../pages/woosync-info-page.php');
}
function render_settings_page() {
    include_once(plugin_dir_path(__FILE__) . '../pages/woosync-settings-page.php');
}
function render_auth_page() {
    include_once(plugin_dir_path(__FILE__) . '../pages/woosync-authentication.php');
}
}
new WooSync();
?>
```

Ezek a függvények a WordPress adminisztrációs oldalak megjelenítéséért felelősek. Nézzük őket részletesebben:

- **render_woosync_page() { ... }**: Ez a függvény betölti és megjeleníti a "woosync-info-page.php" nevű fájlt, amely az adminisztrációs oldal egy részletét tartalmazza. Ennek a fájlnak az útvonala a bővítmény mappájában található.
- **render_info_page() { ... }**: Ez a függvény szintén betölti és megjeleníti a "woosync-info-page.php" nevű fájlt. Mivel ugyanazt az oldalt mutatja be, mint az előző függvény, ezért ugyanazt a fájlt használja.
- **render_settings_page() { ... }**: Ez a függvény betölti és megjeleníti a "woosync-settings-page.php" nevű fájlt, amely a bővítmény beállításainak oldalát tartalmazza az adminisztrációs felületen.
- **render_auth_page() { ... }**: Ez a függvény betölti és megjeleníti a "woosync-authentication.php" nevű fájlt, amely az autentikációs oldalt tartalmazza az adminisztrációs felületen.

Ezek a függvények a bővítmény különböző funkcióinak és beállításainak megjelenítéséért felelősek az adminisztrációs felületen. Minden függvény egy specifikus oldalt jelenít meg, amely a bővítmény működéséhez vagy beállításaihoz kapcsolódik. Az osztály példányosítása (new WooSync();) után ezek a függvények lesznek hozzáférhetőek és hívhatóak a WordPress adminisztrációs felületén keresztül.

woosync-menu.php

```
<div class="woosync-header">
  <div class="woosync-logo">
    
  </div>
  <nav class="woosync-menu">
    <ul>
      <li><a href="?page=woosync-info">Info</a></li>
      <li><a href="?page=woosync-authentication">Authentication</a></li>
      <li><a href="?page=woosync-settings">Settings</a></li>
      <li><a href="https://github.com/orgs/CommerceSync-Hub/repositories"
      target="_blank">Download Desktop</a></li>
      <li><a href="https://github.com/orgs/CommerceSync-Hub/repositories"
      target="_blank"><i class="fab fa-github"></i>GitHub</a></li>
      <li><a href="https://commercesync-hub.github.io/commerce-sync-docs/"
```

```
target="_blank">Doc</a></li>
</ul>
</nav>
</div>
```

Az oldal navigációja látható a fenti kódban. Ezt egy külön fájlba rendeztünk amit a különböző oldalakon a

```
<?php include(__DIR__ . '/../navigation/woosync-menu.php'); ?>
```

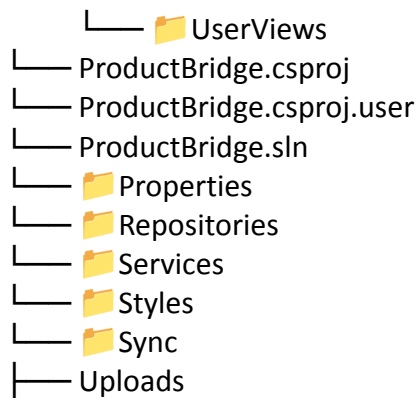
segítségével hívunk meg. Így elegendő és emellett hatékony is a bővítmény saját menüjét csak egy helyen módosítani. A menüpont természetesen rendelkezik a programcsomag logójával, külső és belső linkekkel is.

A PRODUCTBRIDGE ASZTALI ALKALMAZÁS

A ProductBridge Desktop Applikáció segítségével termékeket tölthetünk fel WooCommerce webshopunkba. Elsőként össze kell kötnünk a WordPres-t és a Desktop applikációt. Majd a sikeres autentiikáció után használhatjuk a programot. A következőkben a ProductBridge felépítését és működését tekintjük át.

ProductBridge mappa struktúra

```
├── ProductBridge
│   ├── Assets
│   ├── Commands
│   ├── Controller
│   ├── DataAccess
│   ├── Database
│   ├── MVVM
│   │   ├── Models
│   │   ├── ViewModels
│   │   └── Views
│   │       ├── CredentialViews
│   │       ├── LoginViews
│   │       ├── MainWindow.xaml
│   │       ├── MainWindow.xaml.cs
│   │       ├── MenuViews
│   │       ├── ProductViews
│   │       └── SyncView
```



A ProductBridge a fenti ábrán látható módon épül fel.

Az alkalmazás fejlesztését a Visual Studio IDE segítségével végeztük és igyekeztünk követni az MVVM patternt.

Az MVVM (Model-View-ViewModel) egy tervezési minta, amelyet leggyakrabban alkalmaznak C#-ban és más .NET platformokon, különösen WPF (Windows Presentation Foundation) és UWP (Universal Windows Platform) alkalmazásfejlesztés során. Ez a minta segít elkülöníteni az alkalmazás különböző részeit, javítja a karbantarthatóságot és tesztelhetőséget.

Model (Modell): Ez az alkalmazás üzleti logikáját tartalmazó rész. A modell olyan osztályokat és adatstruktúrákat foglal magában, amelyek az alkalmazás állapotát és viselkedését reprezentálják. A modellnek függetlennek kell lennie a felhasználói felületről és az egyéb megjelenítési rétegektől.

View (Nézet): Ez a felhasználói felületet definiáló rész. A nézetek felelősek az adatok megjelenítéséért és a felhasználói interakció kezeléséért. A nézeteknek minimálisnak kell lenniük az üzleti logikára nézve, inkább csak az adatok megjelenítését kell lefedniük.

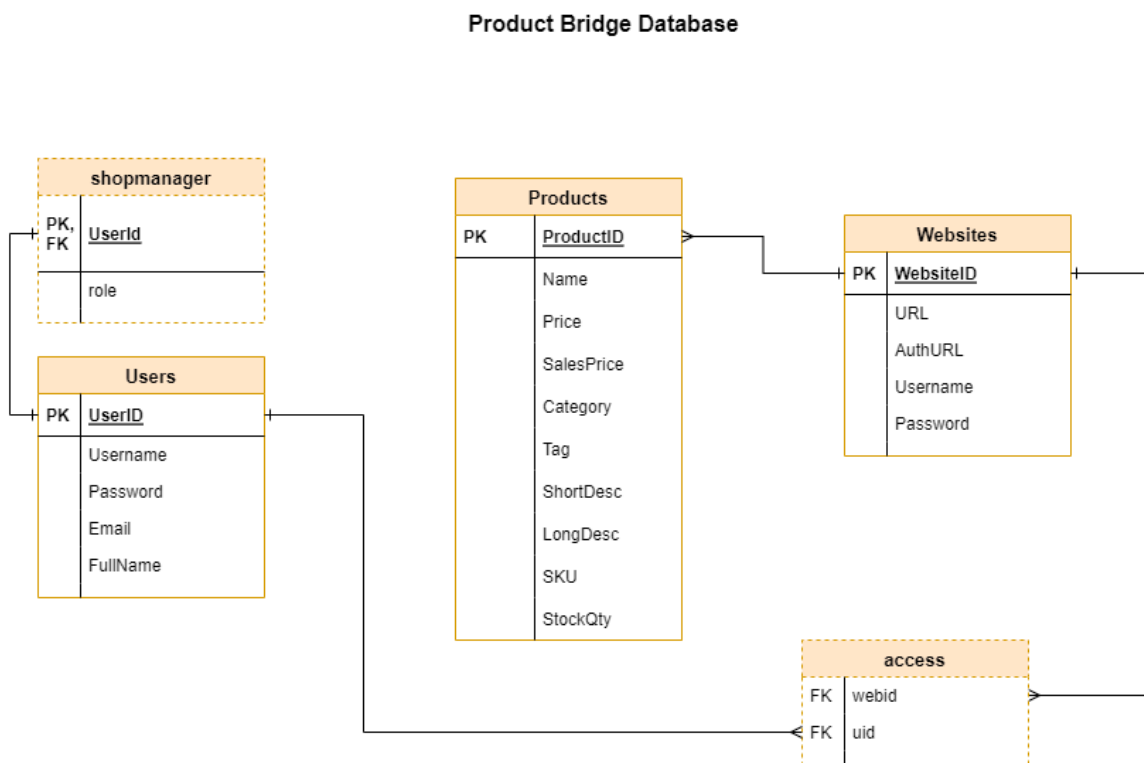
ViewModel (Nézetmodell): A ViewModel a kapcsolódási pont a modell és a nézet között. Ez az osztály tartalmazza az adatokat és a parancsokat, amelyeket a nézet megjelenít. A ViewModel közvetíti a kommunikációt a modell és a nézet között, megkönnyítve a két réteg közötti szétválasztást. A ViewModel továbbá lehetőséget biztosít az adatok formázására és validálására, valamint a felhasználói interakciók kezelésére.

Az MVVM célja, hogy lehetővé tegye az alkalmazás komponenseinek újratervezését és kicserélését anélkül, hogy a többi komponenst érintené. Emellett elősegíti a könnyű tesztelhetőséget, mivel a különálló modell és ViewModel rétegek könnyen tesztelhetők anélkül,

hogy szükség lenne a felhasználói felület bevonására. A tiszta kód és a könnyű karbantarthatóság érdekében fontos betartani az MVVM elveit és eljárásait az alkalmazásfejlesztés során.

Adatbázis

Mindemellett kiemelt feladat volt azt adatbázis tervezése is, hiszen ez adja a szoftver alapját. A következő ábrán láthatjuk az adatbázis tervét, UML diagrammon.



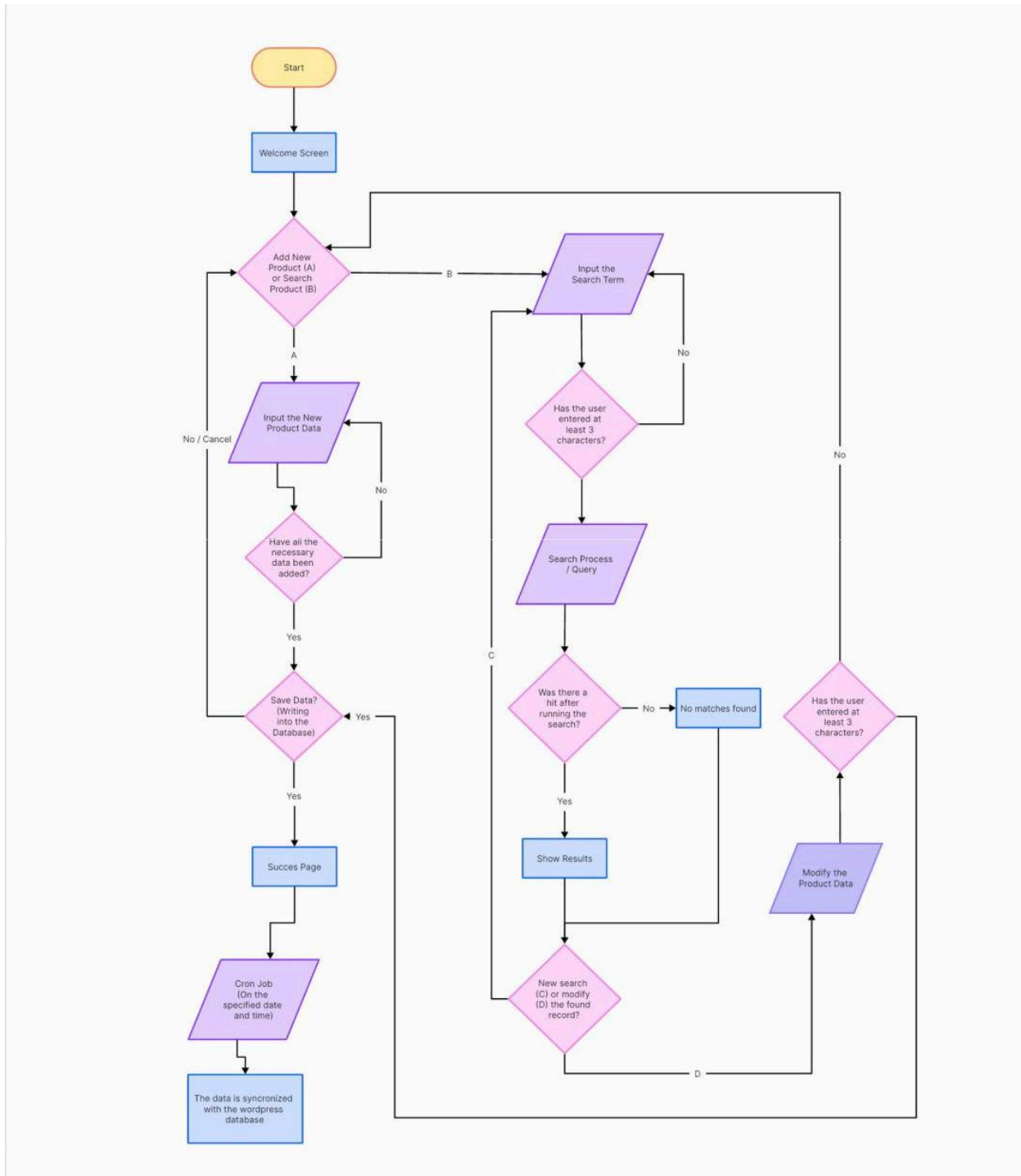
2. ábra: ProductBridge adatbázis tervezet

Egy viszonylag egyszerű adatbázist terveztünk, hiszen az egyszerűség a program célja. Éppen ezért semmilyen felesleges tablát nem tartalmaz a program. Kizárólag azt amire szükség van a működés során.

A következőkben a Desktop alkalmazás működését fogjuk bemutatni. Mindenek előtt a program folyamatábráját terveztük meg, hogy vizualizálhassuk a majdani működést.

A folyamatábrák, vagy más néven diagramok, fontos eszközök a programtervezés során, mivel segítenek vizualizálni és strukturálni a tervezési folyamatot, valamint megérteni és kommunikálni az alkalmazás működését.

Folyamatábra a ProductBridge működéséhez:



3. ábra: A ProductBridge folyamatábrája

A Program alapvetően hét menüpontból áll. Ezek segítségével lehet új terméket hozzáadni, a termékek között keresni, törölni vagy módosítani azokat, felhasználókat keresni, hozzáadni törölni vagy módosítani. A kapcsolathoz szükséges adatokat és kulcsokat megadni. Szinkronizálni. Kilépni a programból.

Assets: Az Assets-ben tároljuk azokat a fájlokat amelyek a megjelenítés során fontosak. Mint például a logó, vagy a login oldal ikonja.

Commands: Ez a fájl egy C# osztályt tartalmaz, amely az ICommand interfészt implementálja. Az ICommand interfész olyan típusokhoz használható, amelyek parancsokat képviselnek, például gombnyomásokat vagy menüelemeket.

A **RelayCommand** osztály egy olyan osztály, amely egyetlen akciót vagy parancsot reprezentál, amely elvégezhető (vagy végrehajtható) egy gombnyomással vagy más eseménnyel. A konstruktorában meg kell adni egy Action<object> típusú paramétert, ami az elvégzendő műveletet tartalmazza, valamint egy opcionális Func<object, bool> típusú paramétert, ami azt határozza meg, hogy az adott művelet végrehajtható-e (a parancs kikapcsolódik vagy sem).

Az osztály két fontos metódust tartalmaz:

CanExecute(object parameter): Ez a metódus meghatározza, hogy a parancs végrehajtható-e az aktuális helyzetben. Ha a canExecute delegált null vagy nincs megadva, akkor mindig igaz értéket ad vissza, ami azt jelenti, hogy a parancs mindig végrehajtható.

Execute(object parameter): Ez a metódus végrehajtja a parancsot, azaz az elvégzendő műveletet.

Ezenkívül az osztály rendelkezik egy RaiseCanExecuteChanged() metódussal is, amelyet akkor kell hívni, amikor a parancs végrehajthatóságát befolyásoló tényezők megváltoznak, hogy a felhasználói felület frissüljön és az új állapotot megjelenítse. Ezáltal az eseményt kiváltó objektum (CanExecuteChanged) meghívásával jelzi az interfésznek, hogy a parancs végrehajthatósága megváltozott, és az interfésznek frissítenie kell az ezt használó vezérlőket.

Controllers: Itt olyan osztályok szerepelnek amelyek a különböző "action"-ökért felelnek, mint például egy gomb lenyomása mögötti logika és műveletek.

DataAccess: Az itt használt kódok adatelérési réteget (data access layer) valósítanak meg, amelyek a MySQL adatbázissal kommunikálnak. A DataAccess osztály több funkciót biztosít az adatbázissal való műveletek végrehajtásához.

- **Konstruktor:** A konstruktor létrehoz egy MySQL adatbázis csatlakozási karakterláncot (connectionString) a megadott szerver, adatbázis, felhasználónév és jelszó alapján.
- **InsertWoosyncAuth() metódus:** Ez a metódus új adatokat szúr be a woosync_auth táblába. Először törli a tábla összes korábbi adatát, majd beszúrja az új adatokat. Paraméterként megkapja az API kulcsot és a webhely URL-t.
- **ResetData() metódus:** Ez a metódus törli az összes táblát az adatbázisból. Ez gyakran használatos tesztelés vagy adatbázis-állapot visszaállítása céljából.
- **GetWoosyncAuth() metódus:** Ez a metódus lekéri a woosync_auth táblából az API kulcsot és a webhely URL-t. Ha talál ilyen rekordot, akkor visszaadja azokat egy tuple formájában, ellenkező esetben üres stringekkel tér vissza.

MVVM: Itt valósulnak meg a nézet, és a nézet mögötti funkciók. Melyeket külön almappákra bontottunk, hogy áttekinthetőbb legyen a kód és a mappastruktúra. A fájlok a mappákon és almappákon belül kapnak helyet.

Például: a SearchProductView.xaml és SearchProductView.xaml.cs.

SearchProductView.xaml.cs:

Ez a kód egy WPF alkalmazásban használt SearchProductView nevű felhasználói felületi részt (view) valósít meg. A SearchProductView egy oldal (Page), amely lehetővé teszi a termékek keresését, megjelenítését és kezelését.

A kód főbb részei:

- **Konstruktor:** Az osztály konstruktora inicializálja a felületet (InitializeComponent() hívás), majd betölti az összes terméket az adatbázisból (LoadProductsFromDatabase() hívás).
- **LoadProductsFromDatabase() metódus:** Ez a metódus betölti az összes terméket az adatbázisból a ProductRepository segítségével, majd beállítja azokat a productListBox listába, ami a felületen jeleníti meg a termékeket. Ha hiba történik a betöltés során, akkor egy hibaüzenet jelenik meg.
- **SearchButton_Click() metódus:**

```
private void SearchButton_Click(object sender, RoutedEventArgs e)
{
    try
    {
        // Keresési feltételek alapján szűrés
        string productName = txtProductName.Text;
        string productId = txtProductID.Text;
        string productSKU = txtProductSKU.Text;
        string productCategory = txtProductCategory.Text;

        // Szűrt termékek kigyűjtése
        var filteredProducts = products.Where(product =>
            product.ProductName.Contains(productName) &&
            product.ProductId.ToString().Contains(productId) &&
            product.SKU.Contains(productSKU) &&
            product.Category.Contains(productCategory)
        ).ToList();

        // Szűrt termékek megjelenítése
        productListBox.ItemsSource = filteredProducts;
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Hiba történt a termékek keresésekor: {ex.Message}");
    }
}
```

Ez a metódus kezeli a keresés gomb lenyomását. Először kinyeri a keresési feltételeket a megfelelő beviteli mezőkből, majd ezek alapján szűri a termékek listáját és frissíti a

productListBox listát a szűrt termékekkel. Ha hiba történik a keresés során, akkor egy hibaüzenet jelenik meg.

- **DeleteButton_Click() metódus:** Ez a metódus kezeli a törlés gomb lenyomását. Kiválasztja a felhasználó által kiválasztott terméket a productListBox listából, majd törli azt az adatbázisból a ProductRepository segítségével. A termék eltávolítása után frissíti a productListBox listát. Ha nem választottak ki terméket a törléshez, akkor figyelmeztető üzenet jelenik meg.
- **ModifyButton_Click() metódus:** Ez a metódus kezeli a módosítás gomb lenyomását. Kiválasztja a felhasználó által kiválasztott terméket a productListBox listából, majd megnyitja az EditProductWindow ablakot a kiválasztott termék módosításához. Ha nem választottak ki terméket a módosításhoz, akkor figyelmeztető üzenet jelenik meg.
- A ProductRepository osztály segítségével valósítja meg az adatbázisba való kommunikációt, és kezeli a termékekkel kapcsolatos adatbázis-műveleteket.

Repositories

Ebben a rétegben a termékek és felhasználókhoz kapcsolódó osztályok találhatóak. Ezek segítenek az adatbázis műveletek végrehajtásában. Például:

A **ProductRepository** osztályt, amely felelős a termékekkel kapcsolatos adatbázis-műveletek végrehajtásáért. A ProductRepository osztály lehetővé teszi a termékek lekérdezését, beszúrását, törlését és frissítését egy MySQL adatbázisban.

A kód főbb részei:

- **Konstruktor:** Az osztály konstruktora létrehozza a MySQL adatbázis csatlakozási karakterláncát a kapott szerver, adatbázis, felhasználónév és jelszó alapján.
- **GetAllProductsFromDatabase() metódus:** Ez a metódus lekéri az összes terméket az adatbázisból a Products táblából, és visszatér egy IEnumerable<Product> kollekcióval, amely tartalmazza az összes termék adatait. A termékek adatait egyenként olvassa ki az adatbázisból, majd létrehozza és hozzáadja egy List<Product> listához.
- **InsertProduct(Product product) metódus:** Ez a metódus új terméket szűr be az adatbázisba a kapott Product objektum adatai alapján. Használja az SQL INSERT INTO parancsot az adatok beszúrásához a Products táblába.

- **DeleteProductFromDatabase(int productId) metódus:**

```
public void DeleteProductFromDatabase(int productId)
{
    using (var connection = new MySqlConnection(connectionString))
    {
        string query = "DELETE FROM Products WHERE ProductId = @ProductId";

        using (var command = new MySqlCommand(query, connection))
        {
            command.Parameters.AddWithValue("@ProductId", productId);

            connection.Open();
            command.ExecuteNonQuery();
        }
    }
}
```

Ez a metódus törli a megadott azonosítójú terméket az adatbázisból. Használja az SQL DELETE FROM parancsot a Products táblában a megadott azonosítójú termék törléséhez.

- **UpdateProduct(Product product) metódus:** Ez a metódus frissíti a termék adatait az adatbázisban a kapott Product objektum adatai alapján. Használja az SQL UPDATE parancsot a Products táblában a megadott azonosítójú termék adatainak frissítéséhez.

Services

Ide a navigációval kapcsolatos Service-t hoztuk létre. Ablakok nyitásával és zárásával foglalkozik a programon belül

Styles

Az applikáció elrendezéseiről, színeiről és általánosságban a megjelenésért felelős fájlok kapnak itt helyet.

Sync

Itt a WooCommerceSync.cs osztály kapott helyet amely felelős termékek és felhasználók szinkronizálásáért egy WooCommerce webáruházhoz. A WooCommerceSync osztály lehetővé teszi termékek és felhasználók küldését a WooCommerce API-jának megfelelő végpontjaira.

A főbb részei:

- **Konstruktor:** A konstruktor kap két paramétert: az API URL-jét és az API kulcsot (apiKey). Ezeket a változókat eltárolja a későbbi használatra.

```
public async Task SyncProducts(List<Product> products)
{
    using (var httpClient = new HttpClient())
    {
        httpClient.DefaultRequestHeaders.Add("Authorization", $"Basic
{Convert.ToBase64String(Encoding.ASCII.GetBytes($"{apiKey}:"))}");

        foreach (var product in products)
        {
            var json = JsonConvert.SerializeObject(product);
            var content = new StringContent(json, Encoding.UTF8, "application/json");
            var response = await httpClient.PostAsync($"{apiUrl}/wp-json/wc/v3/products",
content);

            if (!response.IsSuccessStatusCode)
            {
                throw new Exception($"Failed to sync product {product.ProductName}. Status code:
{response.StatusCode}");
            }
        }
    }
}
```

- **SyncProducts(List<Product> products) metódus:** Ez a metódus felelős a termékek szinkronizálásáért a WooCommerce webáruházhoz. A kapott termékeket végigiterálja, és mindegyiket JSON formátummá alakítja. Ezután HTTP POST kérést küld az apiUrl/wp-json/wc/v3/products címre, hogy feltöltse a termékeket a webáruházba. Ha a válasz nem sikeres (nem 200-as státuszkódú), kivételt dob, amely tartalmazza a hibát.

- **SyncUsers(List<User> users) metódus:** Ez a metódus felelős a felhasználók szinkronizálásáért a WooCommerce webáruházhoz. Hasonlóan működik, mint a SyncProducts metódus, de a felhasználókat küldi el az apiUrl/wp-json/wp/v2/users címre.

Mindkét metódus aszinkron módon fut, ami azt jelenti, hogy az API hívások végrehajtása párhuzamosan történik, anélkül, hogy blokkolnák a fő szálát. Ez lehetővé teszi az alkalmazás számára, hogy továbbra is reagáljon a felhasználói interakciókra, miközben a szinkronizálási folyamat fut.

Uploads

Ebbe a mappába kerülnek a termékekhez feltöltött médiafájlok.

FELHASZNÁLÓI DOKUMENTÁCIÓ

A CommerceSync Bundle egy komplex programcsomag, amely arra hivatott, hogy egyszerűbbé, kényelmesebbé tegye a felhasználók számára a termékfeltöltést WooCommerce webshopok üzemeltetése esetén. A cél, hogy ne akadályokat gördítsünk a felhasználók elé, hanem, hogy megszüntessük az akadályokat. A Felhasználói dokumentáció egy átfogó leírás arról, hogy miként telepíthető és használható a szoftvercsomag.

Rendszerkövetelmények

Böngésző: A legtöbb ma használt böngészőn a WordPress weboldalak legnagyobb százaléka tökéletesen funkcionális és használható. A reszponzív optimalizálásnak köszönhetően, minden kijelzőméreten működni fog az alkalmazás.

Xampp: Az Xampp alkalmazás segít az Apache szerver és az adatbázis futtatásában a számítógépünkön. Ezt mindenképp telepíteni kell. A szoftver letölthető a <https://www.apachefriends.org/download.html> oldalról. A két szoftver indításához szükséges, hogy fusson az Apache és a MySQL.

Windows operációs rendszer: Ajánlott a Windows 10 (64-bit) vagy újabb verziójának használata. Korábbi verziókon a Desktop alkalmazás nem lett tesztelve.

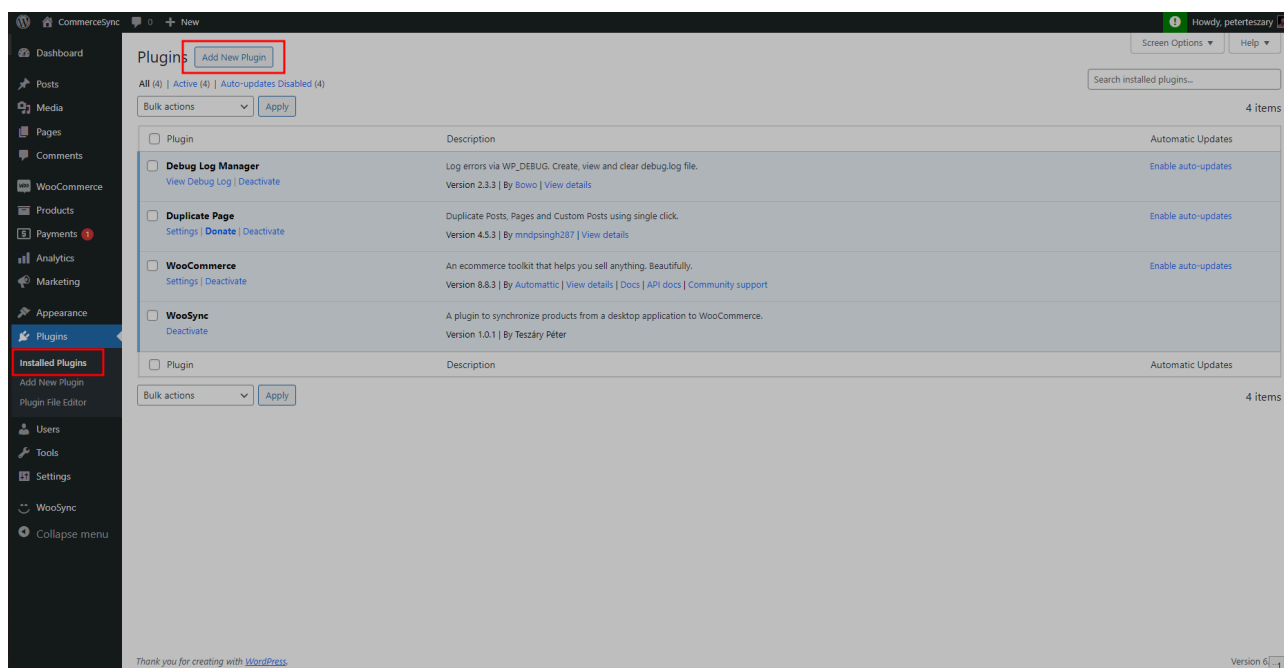
Stabil internet kapcsolat: Az adatok szinkronizálásához elengedhetetlen az internet kapcsolat.

Visual Studio 2022: A program jelenlegi verziója a Visual Studio IDE segítségével futtatható. Ehhez szükséges legalább 80GB tárhely a merevlemezen. Minimum 4GB RAM, Intel Core i5 processzor.

WordPress és WooCommerce: Fontos, hogy a bővítmény használatához szükséges egy már futó WordPress oldal és egy telepített WooCommerce webshop. Ezekhez kapcsolódik a bővítmény, tehát csak ebbe a környezetben tud futni

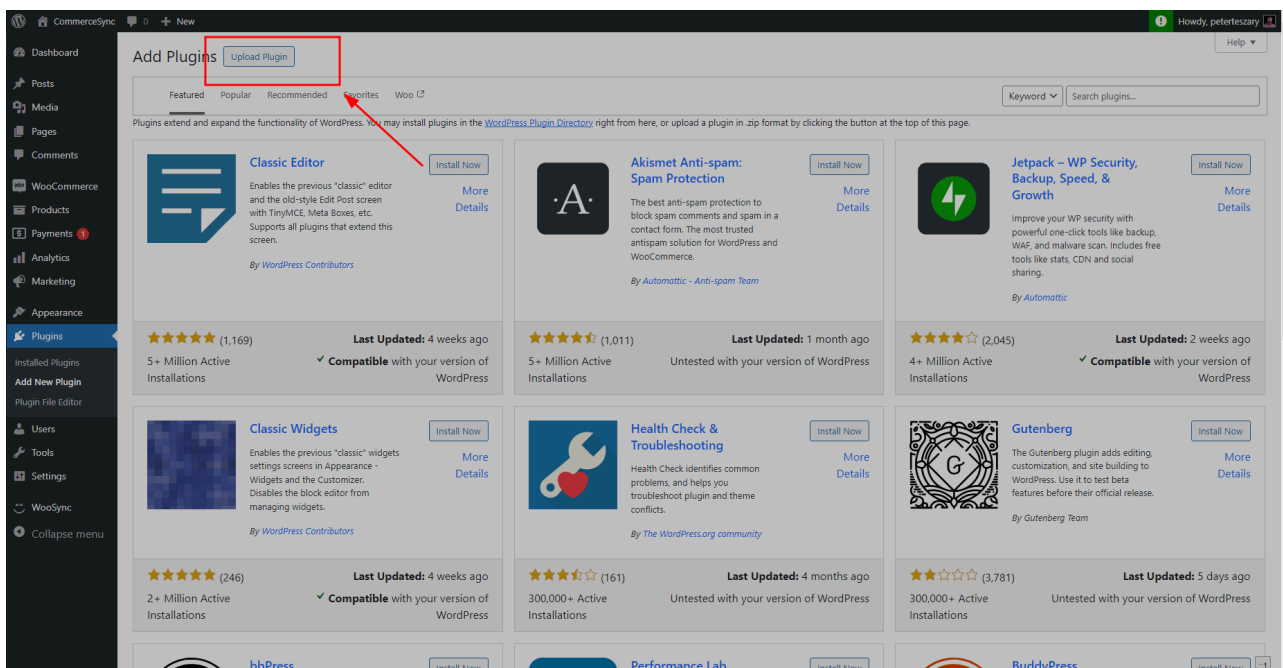
A WooSync bővítmény installálása

A WordPress adminisztrációs felületén navigáljunk a következő felületre **Plugins → Add New Plugin**. Itt kattintsunk a képen látható Add New Plugin gombra.



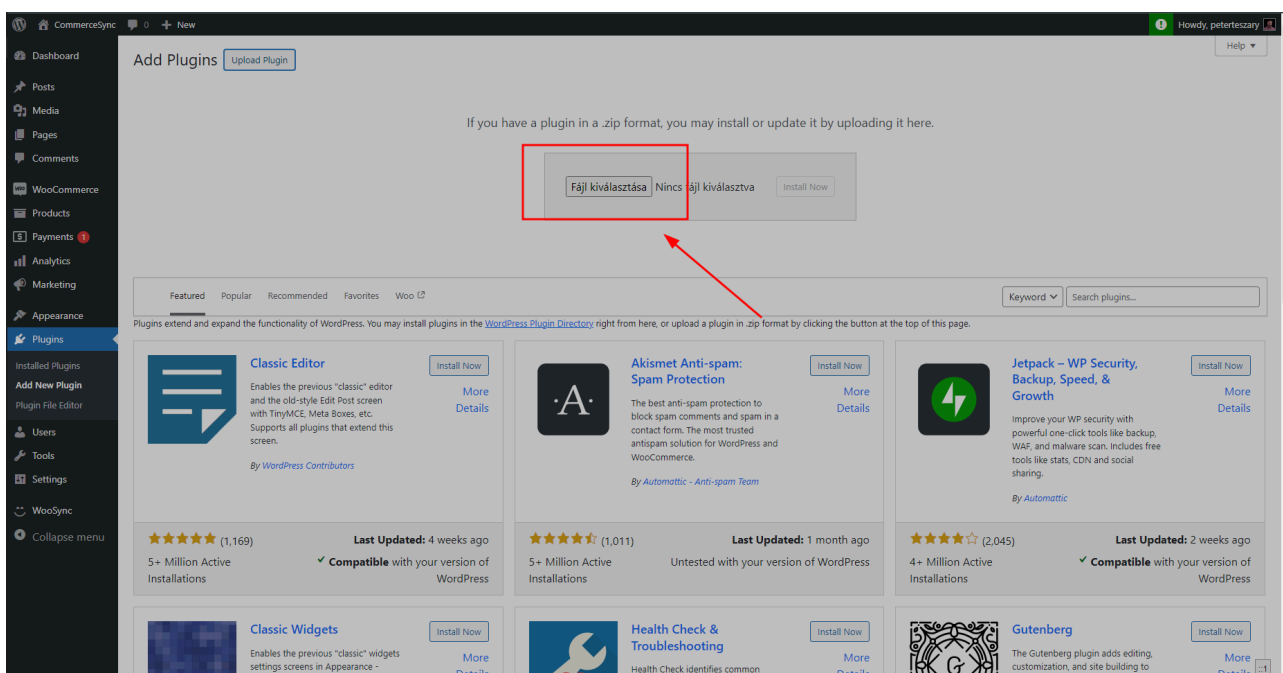
A bővítmény installálása 1.

Válasszuk az **Upload Plugin** opciót a képen látható helyen:



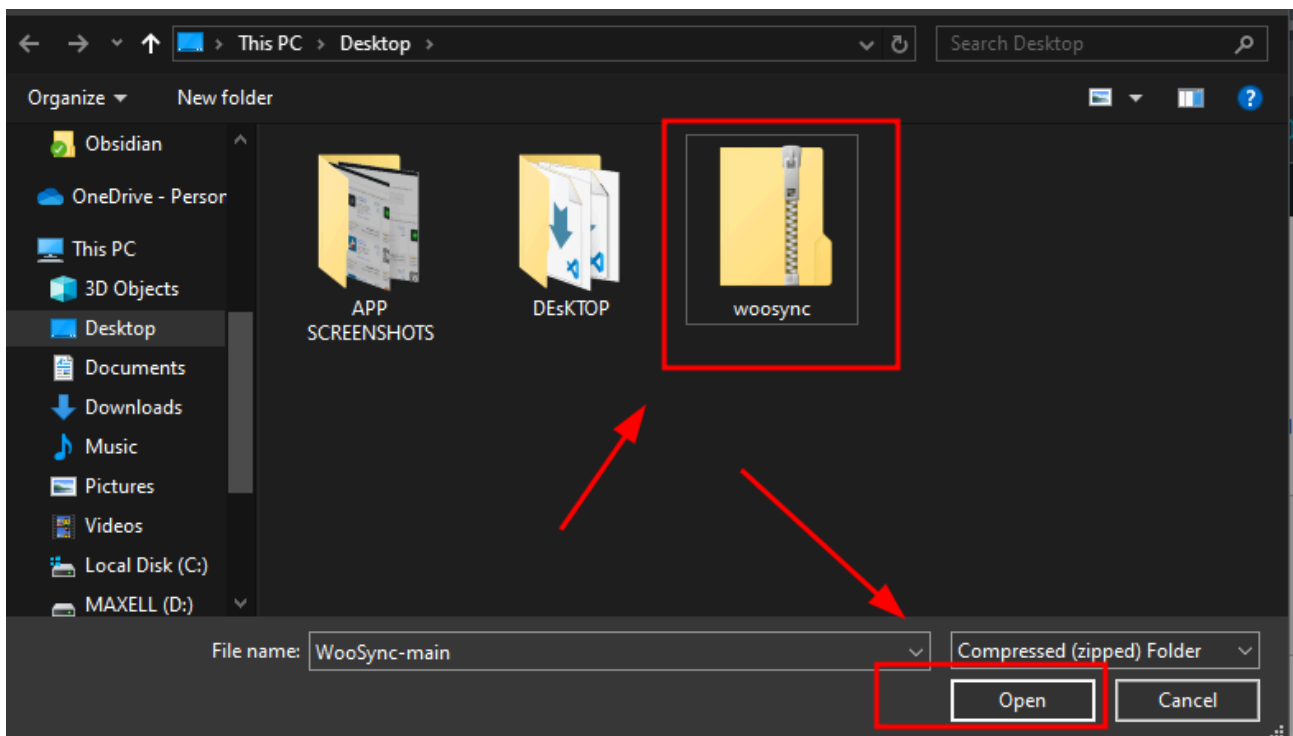
A bővítmény installálása 2.

Kattintsunk a fájl kiválasztása gombra:



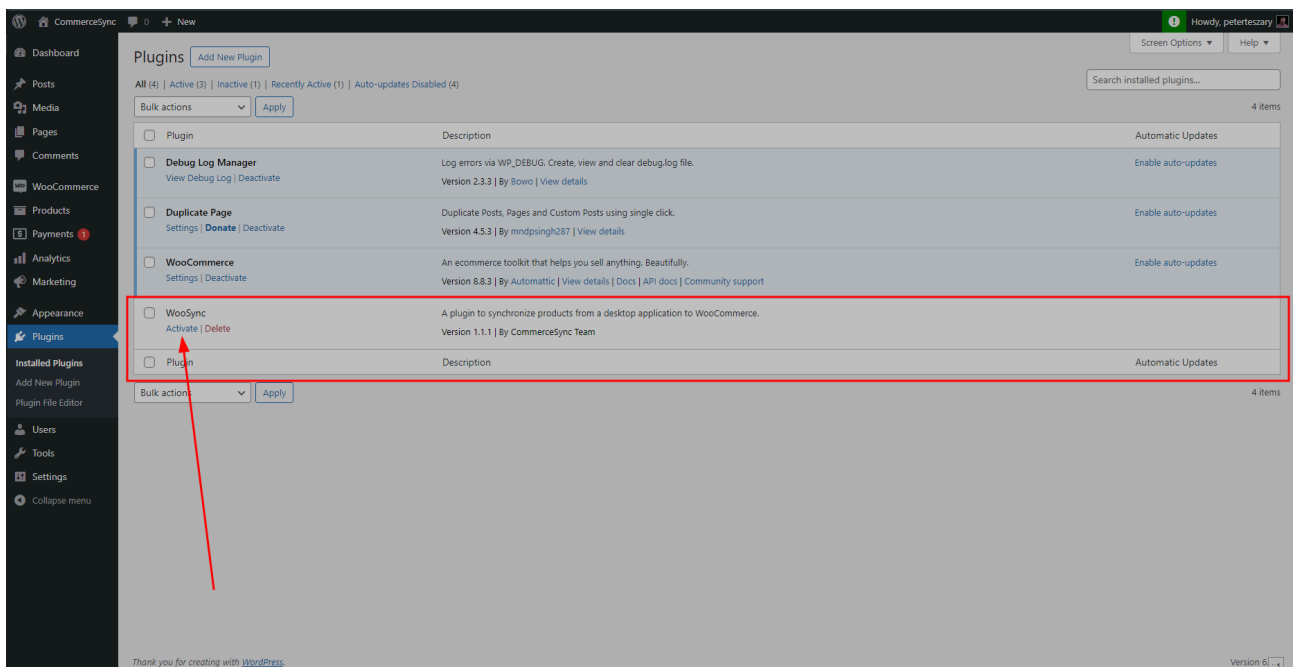
A bővítmény installálása 3.

Tallózzuk be a woosync.zip fájlt:



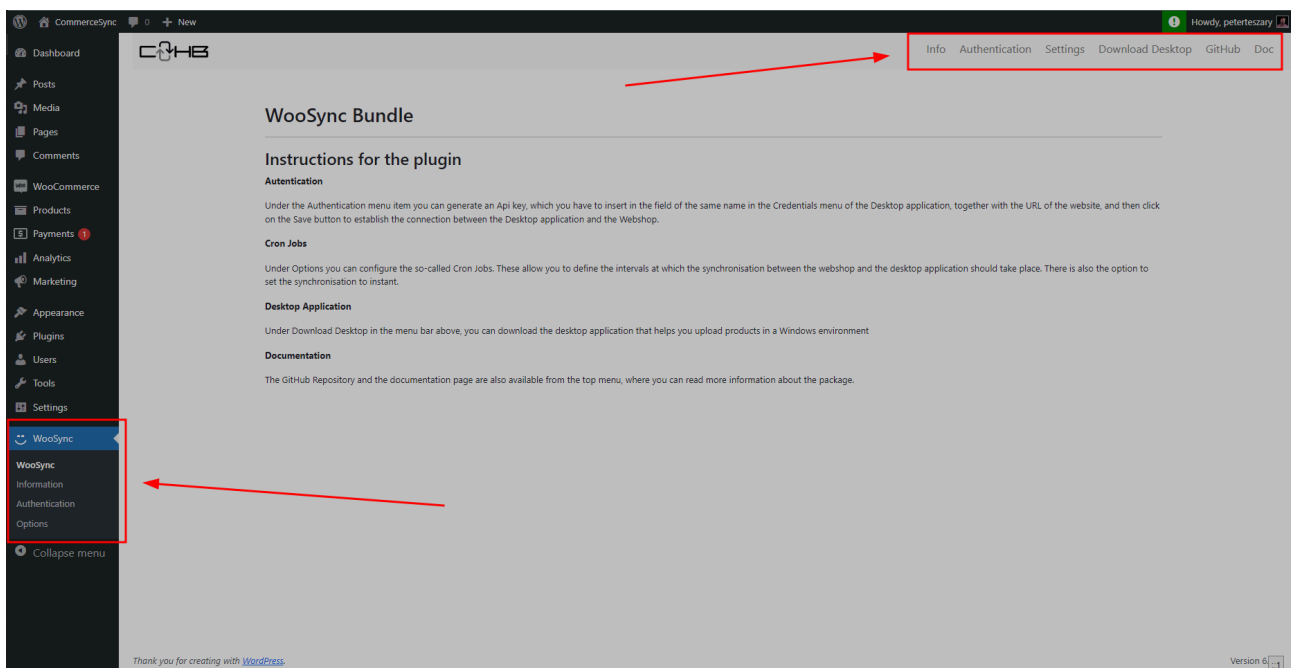
A bővítmény installálása 4.

Aktiváljuk a bővítményt:



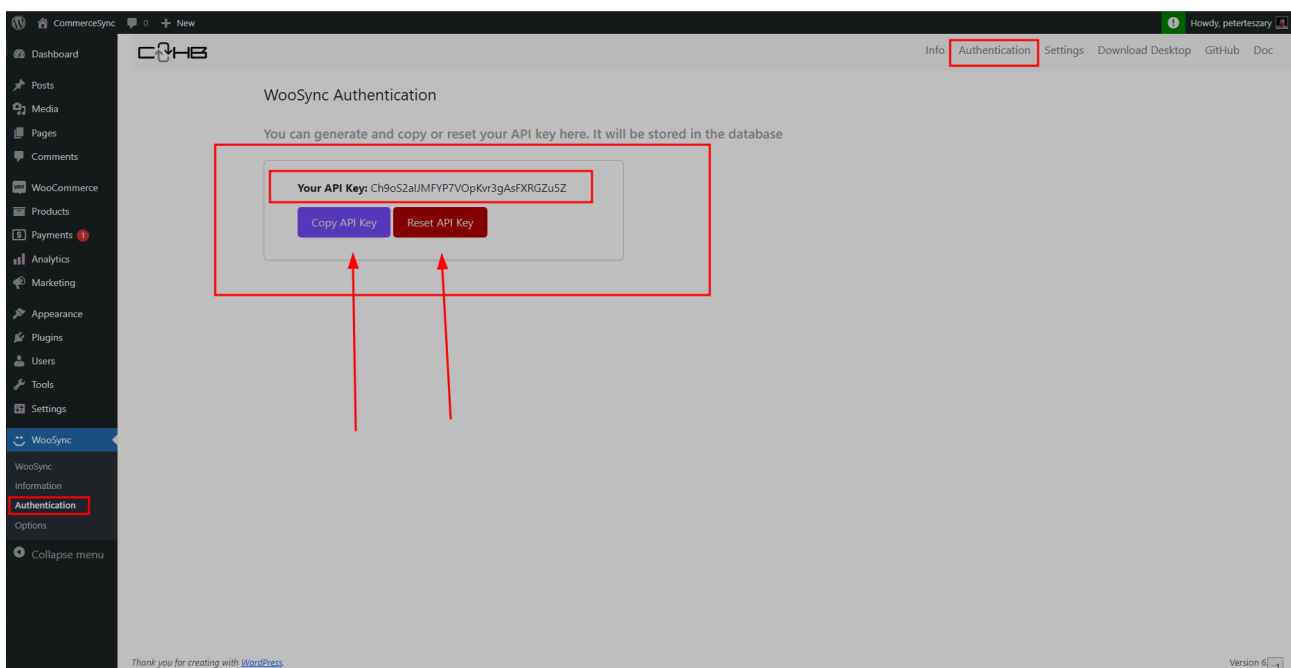
A bővítmény installálása 5.

Az aktiválás után az alábbi módon érhetjük el a menüpontokat:



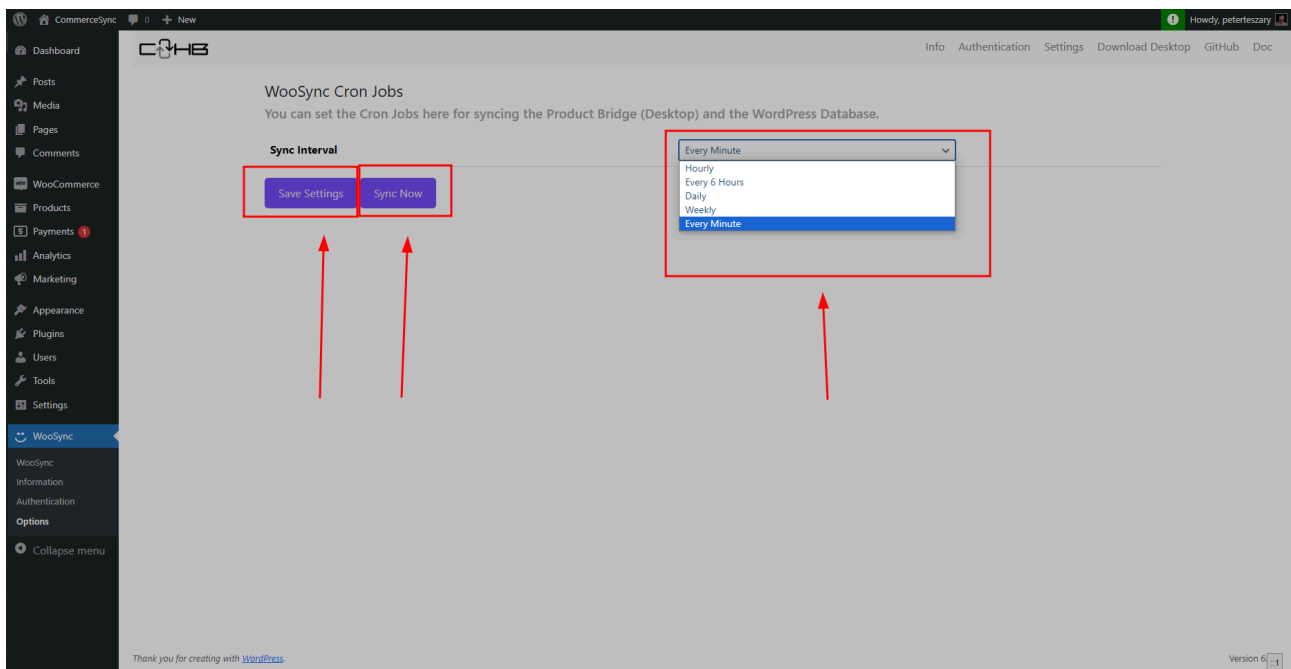
A bővítmény installálása 6.

Az Authentication menüpontban generálhatunk kulcsokat:



A bővítmény installálása 7.

Az alábbi Options menüpontban állíthatjuk be a Cron Job-okat:

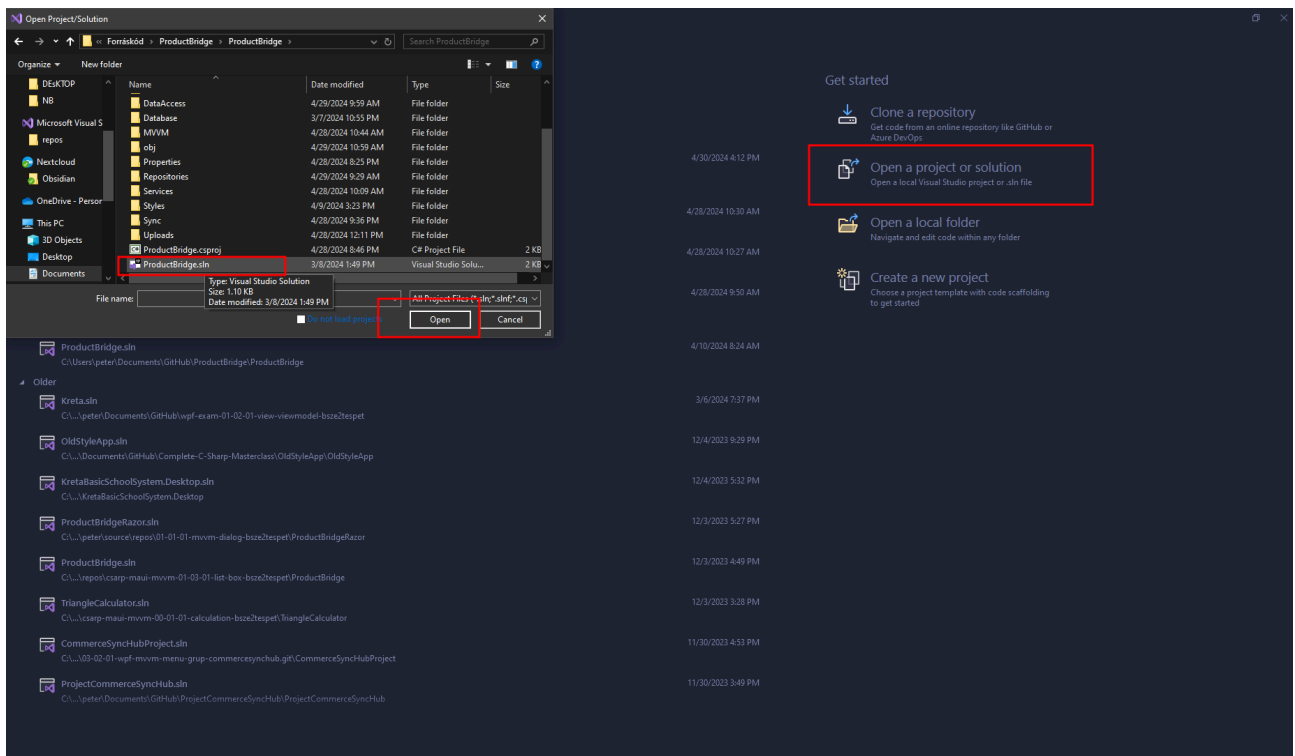


A bővítmény installálása 8.

ProductBridge Desktop alkalmazás installálása

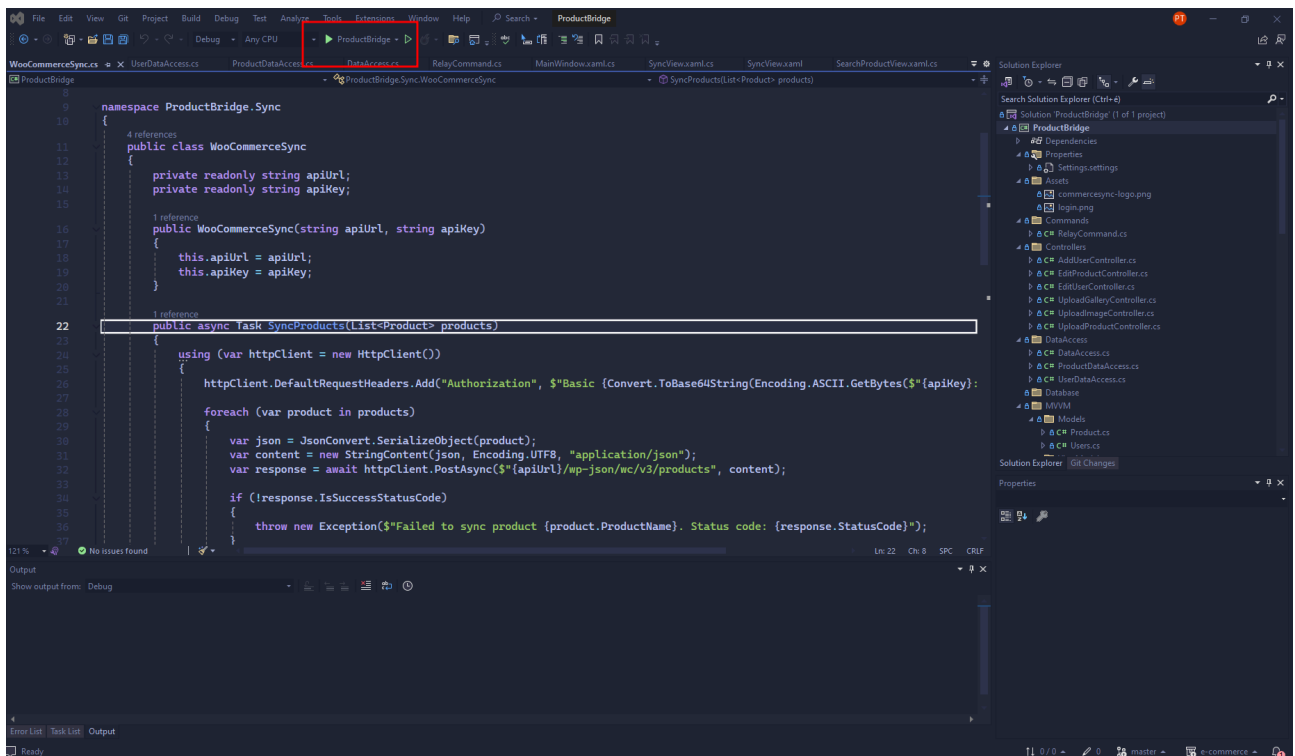
Nyissuk meg a Visual Studio programot.

Tallózzuk be a ProductBridge könyvtárat és nyissuk meg a .sln (solution) kiterjesztésű fájlt:



A Desktop alkalmazás installálása 1.

Futtassuk a programot:



A Desktop alkalmazás installálása 2.

TESZTELÉSI DOKUMENTÁCIÓ

TESZTELÉSI JEGYZŐKÖNYV	
A teszteset leírása és célja:	Képek megjelenítése a dokumentációs oldalon
A tesztelt folyamat / funkció leírása:	A tesztelés során ellenőrizzük, hogy megfelelően megjelennek-e a médiatartalmak a dokumentációs oldalon
A tesztelés előfeltételei:	Működő deployment a Github Pages szolgáltatással
A tesztelés dátuma és ideje:	2024-03-24
A tesztadatok típusa:	Képek visszatérési értéke, megfelelő elérési útvonal
A tesztet végző személy(ek):	Teszárý Péter
A tesztelt rendszer beállításai:	Működő Deployment a GitHub Pages segítségével. Minden funkció működik.
A teszteset elvárt eredménye:	A képek megjelenítése az oldalon
A tesztelés eredménye:	X megfelelt <input type="checkbox"/> nem felelt meg <input type="checkbox"/> megfelelt megjegyzésekkel
Megjegyzések:	Jó néhány tesztelés után megtaláltuk a megoldást. A gyökér mappában kell elhelyeznünk az img mappát és abba a fájlokat. Ezek után már megtalálta a fájlokat, sikeres volt a build és sikeres a deployment is.

A tesztelési jegyzőkönyvet készítette (név, aláírás):	Teszáry Péter
--	---------------

TESZTELÉSI JEGYZŐKÖNYV	
A tesztelés leírása és célja:	Generált kulcsok megfelelő helyre történő mentése
A tesztelt folyamat / funkció leírása:	A tesztelés során ellenőrizzük, hogy megfelelő helyre kerülnek-e a generált kulcsok
A tesztelés előfeltételei:	Működő WordPress weboldal WooCommerce bővítménnyel, futó adatbázissal.
A tesztelés dátuma és ideje:	2024-03-26
A tesztadatok típusa:	<p>php funkció:</p> <pre>function generate_api_key_callback() { \$existing_key = get_option('woosync_api_key'); if (!\$existing_key) { \$api_key = wp_generate_password(32, false); // Save the API key in the database update_option('woosync_api_key', \$api_key); } else { \$api_key = \$existing_key; } }</pre>
A tesztet végző személy(ek):	Teszáry Péter
A tesztelt rendszer beállításai:	Működő WordPress weboldal WooCommerce bővítménnyel, futó adatbázissal.
A tesztelés elvárt eredménye:	Generált kulcsok mentése az adatbázis megfelelő táblájában.
A tesztelés eredménye:	<p>X megfelelt</p> <p><input type="checkbox"/> nem felelt meg</p> <p><input type="checkbox"/> megfelelt megjegyzésekkel</p>

Megjegyzések:	<p>Az alábbi módosítással már működött az adatok írása a megfelelő táblába:</p> <pre>function generate_api_key_callback() { global \$wpdb; \$table_name = \$wpdb->prefix . 'woosync_api_keys'; \$existing_key = \$wpdb->get_var("SELECT api_key FROM \$table_name"); if (!\$existing_key) { \$api_key = wp_generate_password(32, false); \$wpdb->insert(\$table_name, array('api_key' => \$api_key)); } else { \$api_key = \$existing_key; } }</pre>
A tesztelési jegyzőkönyvet készítette (név, aláírás):	Teszárý Péter

TESZTELÉSI JEGYZŐKÖNYV	
A tesztelés leírása és célja:	A felvitt adatok mentése a Desktop Applikációban.
A tesztelt folyamat / funkció leírása:	A tesztelés során ellenőrizzük, hogy mentésre kerülnek-e a rögzített adatok
A tesztelés előfeltételei:	Futó Desktop alkalmazás (ProductBridge) Futó Apache és MySQL
A tesztelés dátuma és ideje:	2024-04-16
A tesztadatok típusa:	Szöveg, szám, decimal

A tesztet végző személy(ek):	Teszáry Péter
A tesztelt rendszer beállításai:	Egyéb beállítás nem szükséges
A tesztet elvárt eredménye:	Az adatokat visszaadja a rendszer a keresés/lista oldalon
A tesztelés eredménye:	<input type="checkbox"/> megfelelt <input type="checkbox"/> nem felelt meg X megfelelt megjegyzésekkel
Megjegyzések:	A hibás teszteket az eredményezte, hogy még mindig a korábbi SQLite adatbázisba mentettük az adatokat, a korábbi beállítások miatt. Ezt kellett módosítani.
A tesztelési jegyzőkönyvet készítette (név, aláírás):	Teszáry Péter

JEGYZÉK

Ábrajegyzék:

1. ábra: Folyamatábra a programok működéséről
2. ábra: A WordPress "Plugins" menüpontja
3. ábra: ProductBridge adatbázis tervezet
4. ábra: A ProductBridge folyamatábrája
5. ábra: A bővítmény installálása 1.
6. ábra: A bővítmény installálása 2.
7. ábra: A bővítmény installálása 3.
8. ábra: A bővítmény installálása 4.
9. ábra: A bővítmény installálása 5.
10. ábra: A bővítmény installálása 6.
11. ábra: A bővítmény installálása 7.
12. ábra: A bővítmény installálása 8.
13. ábra: A Desktop alkalmazás installálása 1.
14. ábra: A Desktop alkalmazás installálása 2.

FELHASZNÁLT IRODALOM

- <https://dataprot.net/statistics/woocommerce-usage-statistics/>
- <https://www.manaferra.com/wordpress-statistics/>
- <https://vitepress.dev/guide/what-is-vitepress>
- <https://www.markdownguide.org/getting-started/>
- <https://developer.wordpress.org/advanced-administration/multisite/create-network/>
- <https://stackoverflow.com/questions/7106012/download-a-single-folder-or-directory-from-a-github-repo>
- <https://www.wpbeginner.com/beginners-guide/beginners-guide-to-wordpress-file-and-directory-structure/>
- <https://developer.wordpress.org/plugins/plugin-basics/header-requirements>

HALLGATÓI NYILATKOZAT

Alulírott (hallgató neve) a Szegedi
Gazdasági Szakképző Iskola Vasvári Pál Tagintézménye hallgatója kijelentem, hogy a
..... (szakdolgozat címe) című záródolgozat a saját
munkám.

Dátum

aláírás