Specifikáció

# Project leírás

## Cél

Játékokat világszerte a legolcsóbb áron eladni, de még úgy, hogy profitot lehessen belőle szerezni. A lehető legnagyobb vásárló központosított ügyfélszolgálatot akarom, egy ticket rendszerrel, vagy közvetlenül e-mailen keresztül tudjanak majd minket elérni.

A marketinget a YouTube videósok és influencereken keresztül kell a legnagyobb részét lebonyolítani. A maradék része Facebook, YouTube, Instagramm hirdetéseken keresztül kell lebonyolítani.

Egy oldal a nagyker eladásokhoz, ami a oem.myLink.hu címen lehetne elérhető. Ezekhez a kódokat ugyanúgy a nagykertől szerezném az elején, és 5%-os profit réssel adnám tovább. Ezzel megkönnyíteném a magyar boltokat az olcsó kódok beszerzésére.

Lehetőség, hogy a felhasználó automatán tudja aktiválni a CD kulcsot, amit vett a saját linkelt Steam accountjára.

Mottó: Ez games Ez Life / Ez Games For Cheap

## Jogosultságok, hatáskörök, project célok

* Több fajta felhasználók, különböző jogkörökkel. Például: tulajdonos, admin, support munkatárs, vásárló
* Két lépcsős azonosítás hozzáadása
* Facebook, Google és ezekhez hasonló regisztrációk biztosítása
* Minden egyes terméket be kell kategorizálni, egy termék több kategóriába is bekerülhet
* Egy hírlevél, ami az akciókról és lehetőségekről add információkat
* Barion, PayPal, bank kártyás, crypto valuta fizetési lehetőség
* A kulcsok azonnali biztosítása a vásárló felé
* Social médiával való összekötésé a rendszernek, még ha csak a marketing része is
* Különböző kupon kódok
* A partnerek is kapnak 1%-ot, ha az ő kuponkódjával vásárólnak, és a vásárló is kap -5%-ot
* Egy Wallet amire feltudnak tölteni pénzt, ha szeretnének, vagy tudnak venni ajándék kártyát, amit kipostázunk, vagy online elérhető, erre a Walletra kapnak pénzt a partnerek, és az Meghivó linkek, és a

## Fogalom meghatározások, rövidítések

**Kulcs** – A termék amit szolgáltatunk, egy szoftvernek az aktivációs kulcsa

**Partner** – Az a személy, vagy cég akivel szerződést kötöttünk, hogy bizonyos juttatások mellé ő reklámozza a céget

**Project** – A webshop, amit ebben a dokumentumban tervezünk meg, és amit a cél megvalósításához használunk

**Tulajdonos** – Az a személy, aki a céget birtokolja, és ő vezeti a teljes webshopot

**Admin** – Az a személy, aki a webshop fejlesztésért, és karbantartásáért felelős

**Support** – Az a személy, aki felveszi a kapcsolatot a problémás ügyfelekkel, és megpróbálja megoldani a gondjukat, a lehető leghamarabb. A partnerekkel ő tartja a kapcsolatot

**Ticket –** Egy levél a munkatársaknak, hogyha valami problémája van a vásárlónak, ezt azon keresztül tudja rendezni

**Vásárló** – Az a személy, aki a profitot termeli, ő a legfontosabb személy, és az ő igényét kell kielégíteni

**Meghivó link –** bárki lehet partner ezzel a Meghívó linkkel, ez egy link, amin ha beregisztrálnak, akkor mindkét tag kap 1 EUR-t a Wallet pénztárcájára

**Wallet –** Ez egy virtuális pénztárca, tudnak rá pénzt feltölteni, és kapnak rá pénzt, ha a PC-vel vagy a Meghivó linkkel vásárol valaki

**PC** – Partner kód, ez az a kód amit egy partner tud megadni magának és, ha ezzel a kóddal vásárolnak akkor úgy fogja a pénzt megkapni a partner, hogy a Walletra fogja neki feltölteni. Egy PC-t bárki létrehozhat

## Használt technológiák

Alkalmazást kiszolgáló - ASP.Net Core 3.1

Adatbázis kiszolgáló szerver - MongoDB és MSSQL 2019

Fejlesztői környezet - MS Visual Studio 2019

Környezet - Docker

Web szerver - IIS Express

OP rendszer - Linux

Domain - eg4a.com

Cache - Redis

Event bus - Azure Service Bus

Nagykereskedés - CodesWholeSale vagy software-codes.com

# Rendszer leírás, megvalósítás

## UI

Az UI-t valószínűleg egy előre elkészített designt fogok megvásárolni. Ezzel a dev sebességet többszörösen felgyorsítom, és előbb tudunk elindulni. Ezzel együtt az UI kinézete, optimalizálása is sokkal jobb lesz, mintha én készítettem volna.

## Funkciók felhasználóknak

### Vásárló

* Regisztráció
* Bejelentkezés
* Achievements szerzése
* Vásárlás
  + Vásárlás megosztása, egy link formájában
  + Vásárlás saját magának
  + Vásárlás ajándékba
    - Egy másik e-mail címre küldi ki a kódot
    - Egy másik profilnak tudja azt elküldeni
* Support ticket küldése
* Profil létrehozása
* Profil módosítása
  + Profil név módosítása
  + Steam felhasználó linkelése
  + Leírás megadása
  + Számlázási cím módosítása
  + Jelszó módosítása
  + Két lépcsős azonosítása bekapcsolása
  + Meghívó link készítése

### Support

* Ugyan az, mint az előbbieknek
* Látja a Support ticketeket, és tud rájuk válaszolni
* Saját vállalati e-mail cím

### Admin

* Ugyan az, mint az előbbieknek
* Support tagokat tud kinevezni

### Tulajdonos

* Ugyan az, mint az előbbieknek
* Látja a statisztikákat, hogy mennyi eladás, és bevétel jött be, melyik volt a legtöbbet használt PC

## Funkciók

### Termék

1. Vásárlás
2. Értékelés
3. Ár változtatás
4. Felfüggesztés, elfogyottra jelölés
5. Raktárát feltölteni kulcsokkal, egy csv fájlból, vagy valami feltöltő gombbal

### Felhasználó

1. Adatmódosítása
2. Két lépcsős bejelentkezés bekapcsolása
3. Törlés
4. Feljebb léptetés
5. Értékelés írása, vásárlás után
6. Support ticket indítása

### Wallet

1. Feltöltés
2. Pénz költés
3. Küldés rá másik felhasználóról
4. Ajándékkártya használata

## Microservices

Az összes microserviceben kell lenni egy ConcurrencyStampnek. Ennek az lesz a szerepe: As the name state, it's used to prevent concurrency update conflict.

For example, there's a UserA named Peter in the database 2 admins open the editor page of UserA, want to update this user.

1. Admin\_1 opened the page, and saw user called Peter.
2. Admin\_2 opened the page, and saw user called Peter (obviously).
3. Admin\_1 updated user name to Tom, and save data. Now UserA in the db named Tom.
4. Admin\_2 updated user name to Thomas, and try to save it.

What would happen if there's no ConcurrencyStamp is Admin\_1's update will be overwritten by Admin\_2's update. But since we have ConcurrencyStamp, when Admin\_1/Admin\_2 loads the page, the stamp is loaded. When updating data this stamp will be changed too. So now step 5 would be system throw exception telling Admin\_2 that this user has already been updated, since he ConcurrencyStamp is different from the one he loaded.

Ezzel meg lesz az akadályozva, hogy ha két admin akar mondjuk árat változtatni, egy terméken és már valaki megtette akkor hibát fog kapni

### Kosár (Cart)

Ebbe a microservicebe kosarakat lehet létrehozni, visszatölteni, elmenteni, törölni. A kosarat felhasználóhoz van linkelve. Ez a microservice a gyors működés miatt a csredis mem cacheben fog futni, nem egy SQL serveren. Itt lesz a megrendelésnek az elindítása, innen fog a megrendelés elkezdődni. Ha egy kosárból eltávolítanak egy terméket, akkor ez nem törlődik, hanem disabledre állítja a tulajdonságát, és még 1 hétig megtartja, azután pedig teljesen törli, és vissza lehet állítani

### SupplierPicker

Ennek a microservicenek a feladata, hogy a megadott beszállítók közül kiválasztja azt, amelyik a legjobb árral rendelkezik, és visszaadja azt az URL-t/ adatot, amit az Order szervice megtudja vásárolni a terméket. Ez a service 1 percenként frissíti a legjobb árakat, és a Reddis for .NET memória cachinget fogja használni, és onnan fogja a queryket kiszolgálni.

### Marketing

Ebbe a microservicebe a marketinget lehet kezelni. Ez annyit jelent, hogy innen lehet hírleveleket kiküldeni, ez alapján fogja az ajánlott játékokat kiküldeni.

### Identity

Ezen a microservicen keresztül fogja a rendszer azonosítani a felhasználókat. Itt lehet kezelni a profilokat, és a profilok adatait. Az azonosítás JWT token keresztül fog menni.

### Katalógus (Catalog)

Ezen keresztül lehet a termékeket karbantartani, bővíteni, árat csökkenteni, discountot adni. A katalógusban kell lennie egy Reddis memória cachenek, ami a lekérdezett termékeket cacheli.

### Rendelés (Ordering)

Ezen a microservicen keresztül fognak a rendelések végig futni. Ez a legfontosabb microservice, ezt kell a legbiztonságosabban megtervezni.

### Támogatás (Support)

Ezen a microservicen keresztül fog a support ticket rendszer működni, és itt lehet válaszolni a ticketekre, ticket írni.

### Pénztárca (Wallet)

Ezen a microservicen keresztül lehet kezelni a pénztárcát. A pénztárca event sourced alapon fog működni. Tehát 0-val fog indulni és minden egyes tranzakció számon lesz tartva SQL adatbázisba.

## Rendszer leírás

A belső microservicek között gRPC kommunikáció és RabbitMQ vagy Azure Service Bus Event Bus fog futni, ezen keresztül a rendszerben felküldők eventeket, minden adattal ami kell, és ez fogja irányítani a többi service adatait is. Az authentikáció a microservicek között egy JWT tokkennel megy, ami tartalmazza a felhasználónevet, e-mail címet, és a Roles-okat, ez lesz a token tartalma. A kliens és szerver kommunikáció között pedig REST API fog futtni, egy Api Gatewayel és MVC modellel.

Lehetőség lesz elérni egy Server Side Blazor appot is, ami a lehető legkevesebb erőforrást igényel, ezért az folytonos használatra, gyenge gépeken tökéletes. A Mobil applikációt is a Server side vagy a WebAssambly Blazor fogja ellátni.

## JWT Token, és annak tartalma

A JWT Tokennen keresztül fog működni az autentikáció a microservicek között. A JWT token tartalmazni fogja a Rolesokat, ami szükséges az adott feladat végrehajtásához. A Token a payloadban tartalmazni fogja a tokent lekérő IP címét, és minden egyes azonosításnál azt megfogja nézni az azonosító rendszer, hogy a küldő IP cím azonos e a payloadban szereplővel

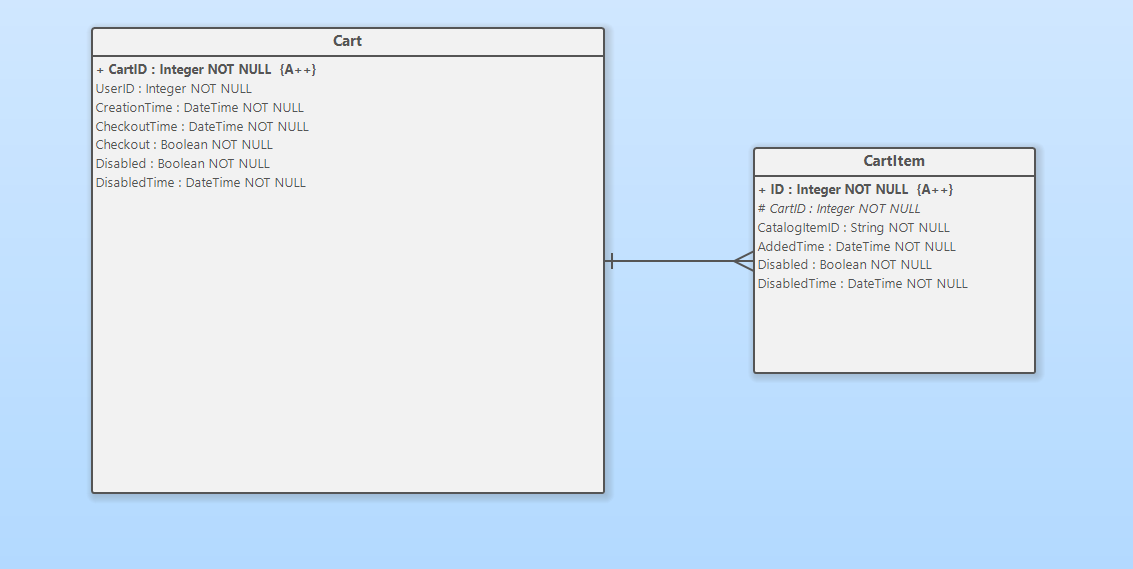
### Tartalma

* Username
* E-Mail
* UserID
* Keresztnév
* Vezetéknév
* Roles
* IP cím
* Nyelv pl (HU\_hu)
* Currency pl (HUF)

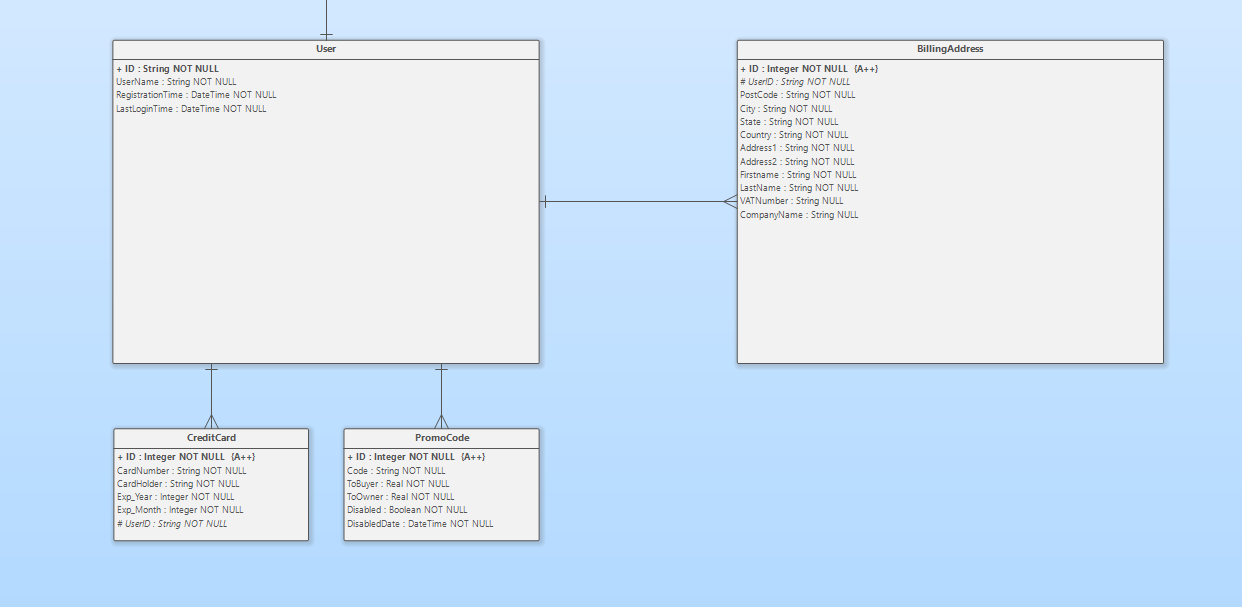
## Adatbázis

### ER model

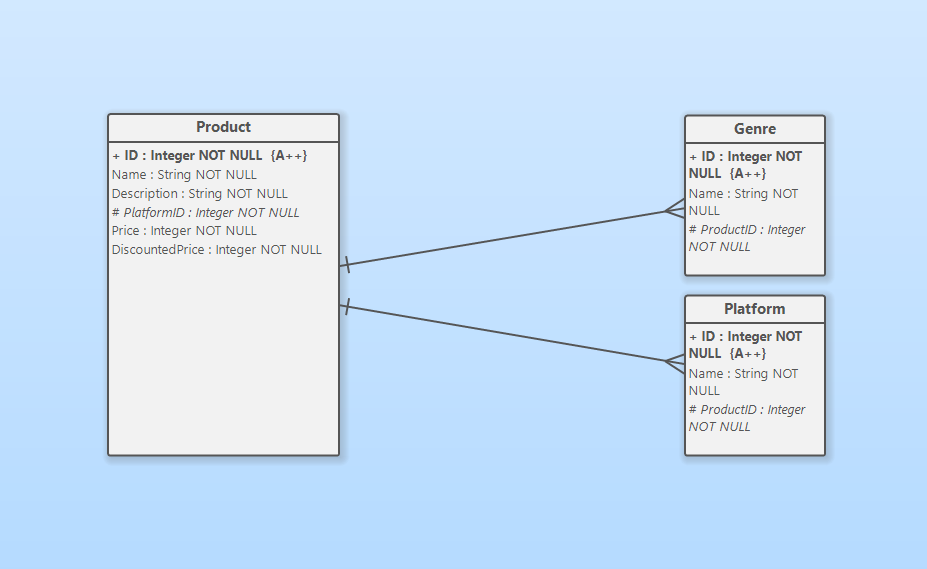
#### Cart



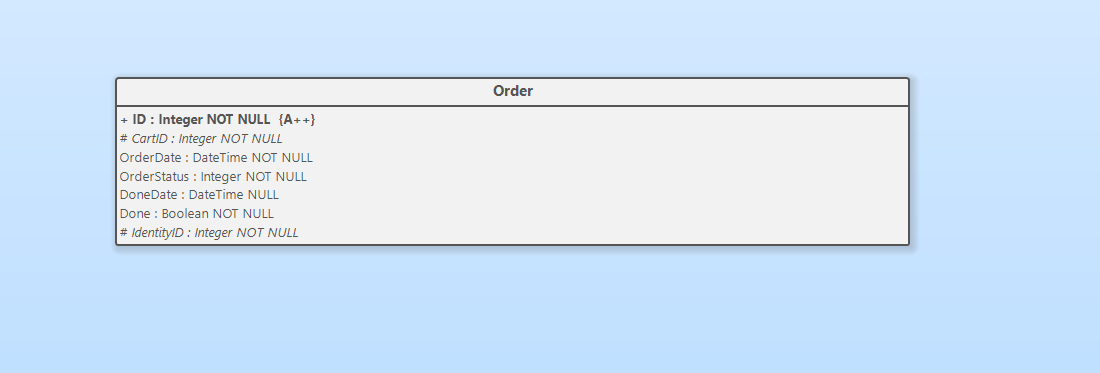
#### Identity



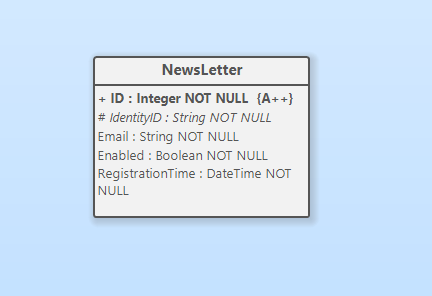
#### Catalog



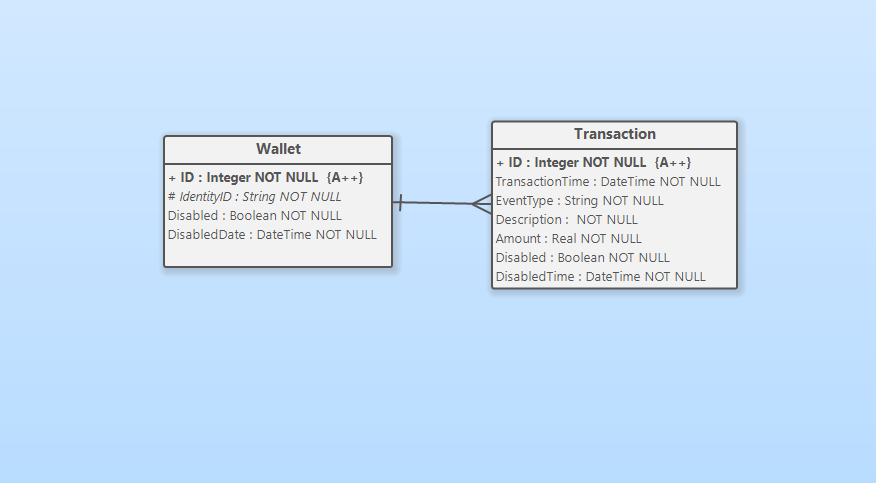
#### Order



#### Marketing



#### Wallet



## Work flow

### Regisztrálás

A felhasználó rákattint a regisztrálás gombra. A rendszer elnavigál a /registration lapra. A felhasználó kitölti az adatokat és beküldi a formot. A rendszer megnézi az adatok validságát, ha nem jó visszaküld egy hibakódot. Ha jó akkor feltölti az adatbázisba és visszaküld, és bejelentkezteti a felhasználót. A regisztrálás után a rendszer kiküld egy Registration eventet az eventbusra, az adatok: UserID, Time, E-Mail.

#### Workflow

### Bejelentkezés

A felhasználó rákattint a bejelentkezés gombra, a rendszer elnavigálja a /login lapra. A felhasználó kitölti az adatait és a rendszer beküldi a formot. A rendszer megvizsgálja az adatok validságát, és ha nem jó akkor visszadob egy hibát. Ha jó, akkor a rendszer tovább megy, és megnézi, hogy felhasználó bekapcsolta-e a két lépcsős azonosítást, ha igen, akkor tovább dobja a /2fact lapra, a felhasználó megadott adataival, amint sikeresen megcsinálta a 2 lépcsős azonosítást, akkor a felhasználó kap egy JWT tokent.

#### Workflow

# Részletes leírás

## Identity microservice

### Endpoints

/user/register

/user/login