1. Egy kis vegyesbolt vezetője felkéri, hogy készítsen egy számítógépes programot, amely a bolt raktárkészletét tartja nyilván. Határozza meg a fejlesztendő szoftver funkcionális és minőségi követelményeit, valamint az alkalmazás szerkezeti felépítését!

-Mutassa be a programtervezés folyamatát!

Egy vegyesbolt árukészletéről van szó, és mivel egy termékből jó esetben nem csak egy darab van, egy adatbázis mindenképpen kell. Ezt az adatbázis az egyszerűség kedvéért nevezzük el leltar.sql. Az adatbázisnak tartalmaznia kell a termék azonosítóját, nevét, fajtáját (mert hát mondjuk nem mindegy, hogy egy termék étel vagy vegyszer), tömegét, árát, és darabszámát. Ezek legyenek mondjuk legyenek a következők: „id", „nev", „fajta", „tomeg", „db". Ezeknek az adatait xml formátumba érdemes tenni, az szerkesztés egyszerűségének kedvéért pontosvesszővel elválasztva. Ezt érdemes letesztelni, ha nem vagyunk biztosak egy-egy parancssor működésében, erre a legegyszerűbb a webalapú ellenőrzés az ingyenesen letölthető XAMMP-al, ami átirányít a localhostra és a PHPmyadmin felületére, amin van MySQL lehetőség is. Azon kívül, hogy egyszerűbb ez a megoldás, még az is a javára válik, hogy operációsrendszerfüggetlen, lényegében így mindegy, hogy Windows, MACOS, Linux vagy Raspberry PI van a gépre telepítve. A párosítás megoldható C-típusú vagy egyéb programozási nyelvek esetében is (pl. Python-ban vagy Java-ban).

-Nevezze meg és jellemezze a tervezés során készítendő dokumentumokat!

- leltar.xml
- vegyesbolt.sln
    - program.cs
- vegyesbolt.form

-Határozza meg a fejlesztendő szoftver funkcionális követelményeit!

- Lehessen manuálisan változtatni a darabszám értékét

-Határozza meg a fejlesztendő szoftver minőségi követelményeit!

- CLI és GUI formátumban is legyen megfelelően átlátható

CLI (lekérdezések során) példa:

1-Melyik a legolcsóbb termék?

Vanília aroma, 30g, 120Ft

2-Miből van a legtöbb?

Gyermelyi cérnatészta, 10db, 80g, 600Ft

STB.

Kidolgozó: Bubla Levente

Validáló: Demsa Attila Valentin

2. Egy desktop ügyviteli szoftver tesztelését kell megvalósítani. Készítse el a tesztelési tervet, tegyen javaslatot a tesztelési környezetkialakítására!

Definiálja a tesztelés fogalmát és jellemzőit!

Tesztelés: A tesztelés a szoftvertermékek minőségének ellenőrzése. A tesztelés során a tesztelők a szoftvertermék képességeit vizsgálják, és megállapítják, hogy a termék megfelele a követelményeknek.

Határozza meg a szükséges teszttípusokat!

1. **Funkcionális tesztelés**: A funkcionális tesztelés a szoftvertermék képességeit vizsgálja. A tesztelők megállapítják, hogy a termék elvégzi-e a kívánt funkciókat, és megfelel-e a

követelményeknek.

2. **Teljesítménytesztelés**: A teljesítménytesztelés a szoftvertermék sebességét, hatékonyságát és állandóságát vizsgálja. A tesztelők megállapítják, hogy a termék megfelel-e a követelményeknek, és hogy a termék megfelelő sebességgel és hatékonysággal működike.

Határozza meg a tesztelés szintjeit!

1. **Alapvető tesztelés**: Az alapvető tesztelés a szoftvertermék képességeit vizsgálja. A tesztelők megállapítják, hogy a termék elvégzi-e a kívánt funkciókat, és megfelel-e a

követelményeknek.

2. **Középszintű tesztelés**: A középszintű tesztelés a szoftvertermék képességeit vizsgálja, és megállapítja, hogy a termék megfelel-e a követelményeknek. A tesztelők továbbá megvizsgálják a termék kompatibilitását más szoftverekkel, és ellenőrzik, hogy a termék megfelel-e a követelményeknek.

3. **Magas szintű tesztelés**: A magas szintű tesztelés a szoftvertermék képességeit vizsgálja, és megállapítja, hogy a termék megfelel-e a követelményeknek. A tesztelők továbbá megvizsgálják a termék kompatibilitását más szoftverekkel, és ellenőrzik, hogy a termék megfelel-e a követelményeknek. A tesztelők továbbá ellenőrzik a termék biztonságát és megbízhatóságát.

Jellemezze a tesztelési módszereket!

1. **Unit tesztelés**: A unit tesztelés a szoftvertermék egy egységét vizsgálja. A tesztelők megállapítják, hogy az egység megfelel-e a követelményeknek, és hogy a termék megfelelően működik-e.

2. **Integrációs tesztelés**: Az integrációs tesztelés a szoftvertermék különböző egységeinek együttműködését vizsgálja. A tesztelők megállapítják, hogy a termék megfelel-e a követelményeknek, és hogy a termék megfelelően működik-e.

3. **Regressziós tesztelés**: A regressziós tesztelés a szoftvertermék korábbi verzióinak tesztelését vizsgálja. A tesztelők megállapítják, hogy a termék megfelel-e a

követelményeknek, és hogy a termék megfelelően működik-e.

4. **Elfogadási tesztelés**: Az elfogadási tesztelés a szoftvertermék képességeit vizsgálja. A tesztelők megállapítják, hogy a termék megfelel-e a követelményeknek, és hogy a termék megfelelően működik-e.

Definiáljon egy teljes tesztelési környezetet!

A teljes tesztelési környezet a szoftvertermék teszteléséhez szükséges eszközöket, módszereket és folyamatokat tartalmazza. A teljes tesztelési környezet magában foglalja a következőket:

1. **Tesztelési eszközök**: A tesztelési eszközök a szoftvertermék teszteléséhez szükséges eszközöket tartalmazzák. A tesztelési eszközök közé tartoznak a tesztelési szkriptek, a tesztelési adatbázisok és a tesztelési segédprogramok.

2. **Tesztelési módszerek**: A tesztelési módszerek a szoftvertermék teszteléséhez szükséges módszereket tartalmazzák. A tesztelési módszerek közé tartoznak a funkcionális tesztelés, a teljesítménytesztelés, a kompatibilitástesztelés és az elfogadási tesztelés.

3. **Tesztelési folyamatok**: A tesztelési folyamatok a szoftvertermék teszteléséhez szükséges folyamatokat tartalmazzák. A tesztelési folyamatok közé tartoznak a tesztelési tervezés, a

tesztelés végrehajtása és a teszteredmények értékelése.

3. tétel

Ebben a tételben a játékfejlesztéssel járó problémákat, a főbb verziókezelő rendszereket, illetve a programváltozatok frissítését fogom kifejteni.

Saját fejlesztése során egy programozó csapatnak rengeteg problémával kell szembenéznie. A legelső, hogy a csapat nem megfelelő fejlesztői környezetet választ, az általuk elképzelt játékuknak. Minden motornak van egy bizonyos korlátozása, amit már nem képes végrehajtani. A második probléma lehet a játék helyes optimalizációja, hogy a kód minél kevesebb erőforrást használjon, de emellett jól használja a ki a hardver adta lehetőségeket. A harmadik ilyen probléma a kódban elkövetett hibák, más néven BUGOK. Nem ezeknek a javítása okoz problémát a programozóknak, hanem a megkeresésük a több 100.000 sorból.

Játékot keretrendszerrel vagy anélkül tudják fejleszteni a programozók. Manapság már kevésbe népszerű a keretrendszer nélküli programozás, annak a bonyolultsága miatt. Ilyenkor a játékot a nulláról kell felépíteni, főképp alacsonyszintű programozási nyelvet használva, mint például C++. A megjelenítéséhez pedig API-kat használnak, megkönnyítik ezzel a programozók feladatát. Ezek a programrendszer lehet pl. OpenGL, segít a 3D-s grafikában. 0-ról kell felépíteni a programot, így a megejtett hibákat a kódban könnyebben kitudják javítani. Kezdő játékfejlesztőknek ez a megoldás általában hátrány lehet, a fejlesztési folyamat sokkal időigénysebbé válik.

Keretrendszereket a programozók a 2000 körüli években kezdték el használni. Két részre tudjuk szedni ezeket. Grafikusmotor, ami csak a képi megjelenítésben segíti a programozót. A másik pedig a játékmotor, ami nagyobb eszközteret biztosít nekik. Az utóbbi a népszerűbb, mivel tartalmaz különböző alrendszereket, például a bevilágítás, fizika, mesterséges intelligencia. Hátránya a magas ár, kevesebb rugalmasság, szabadság és nehezebb kijavítani a hibákat benne, viszont gyorsabb a bennük lévő fejlesztés, mint a keretrendszer nélküliben. A mai legismertebb motorok a Unity, Unreal Engine, CryEngine, Source.

A legelső fázis a játékfejlesztés során az ötletelés és az alapötlet pontos meghatározása. Ez után következik, hogy milyen platforma szeretnék fejleszteni a játékot. Lehet ez PC, web, konzol, telefon és főképp ez fogja meghatározni a játékmotort is. Ha ezekből minden teljesült, a fejlesztők nekiállhatnak egy prototípust gyártani, ami tartalmazza az alapmechanikákat. Ilyenkor még a játéknak nincsen designja, csak egyszerű szürke grafikája van. Az a célja, hogy kialakuljon a végső design, a játékos cselekedetei, illetve a játékszabályok. Ilyenkor egy elég hosszú dokumentumot kell készíteni a fejlesztésről, leírni a játékmenetet és milyen alapvető változásokat kéne meghozni. Következő fázis a leghosszabb, úgy hívják gyártás. Ilyenkor már nem történik nagy változtatás, a fejlesztők megkapják a feladatukat és annak a határidejét. Részletes dokumentum alapján megcsinálják a grafikai asseteket és ezzel párhuzamosan pedig előhívják a játékmenetet, szép lassan elkészül a játék. Ennél a résznél a játékosok már láthatnak bemutatót a közelgő alkotásról és ilyenkor kezdi marketinget a kiadó. A negyedik fázis a karbantartás, a

megjelenést követő hibák javítása. A fejlesztők akár későbbi kiegészítőket is csinálhatnak a meglévő játékmenethez, úgynevezett DLC-ket.

A játékszoftverek életciklusát nagyban meghatározza a játéksok érdeklődése az adott játékkal kapcsolatban. Egy átlagos játékot a kiadó akár 2-3 évig is támogat, ezt főképp az előbb említett dolog határoz meg. Azért is kell ennyi idő, hogy a fejlesztés és a marketing árát teljesen visszahozza a játék és profitálni is tudjon a kiadó. Rengeteg ember használ kalózjátékokat, amivel megkárosítja a bevételtől.

A frissítések a felhasználói visszajelzésekből jönnek létre. A játékosok tudomást adnak a fejlesztőknek a hibákról. Ilyenkor a csapat megpróbálják kijavítani és biztosítani a megfelelő játékélményt a játékosok számára. Frissítések lehetnek még játékmenet újítások, hogy minél több ideig lehessen fent tartani az érdeklődést a program körül.

4. Webalapú rendszerek felhasználókezelési megoldásai és jellemzői

- Felhasználói fiókok létrehozása és kezelése:

A webalapú rendszerek felhasználókezelési megoldásai különböző módokon valósíthatók meg. Ezek a megoldások lehetnek:

1. Az alkalmazás bejelentkezési oldalának használata: Ez az elterjedtebb módszer, amely lehetővé teszi a felhasználók számára, hogy biztonságosan belépjenek az alkalmazásba. A felhasználónak meg kell adnia egy hitelesítő adatot (például e-mail címet vagy felhasználónév és jelszó párost), amelyet az alkalmazás ellenőrizni fog. Ha a hitelesítés sikeres, akkor a felhasználónak hozzá kell férnie az alkalmazás funkcióihoz.

2. OAuth hitelesítés: Ez egy olyan technológia, amely lehetővé teszi a felhasználóknak, hogy biztonságosan csatlakozzanak egy harmadik féltől származó hitelesítő szolgáltatásokhoz (például Google vagy Facebook). A felhasználónak csak egyszer kell bejelentkeznie a harmadik féltől származó szolgáltatásba, és utána automatikusan hozzáférhet az alkalmazás funkcióihoz.

3. SSO (Single Sign-On): Ez egy olyan technológia, amely lehetővé teszi a felhasználónak, hogy egyszerre jelentkezzen be több webhelyre anélkül, hogy minden helyszínen újra meg kellene adnia a hitelesítő adatait. A SSO segítséget nyújt abban is, hogy meghatározza és kontroll alatt tartja a felhasználói jogosultsági információkat is.

- Jelszavak kezelése:

A felhasználók a webes rendszereken jelszavakat hozhatnak létre, és azokat később kezelhetik. Így különböző jelszavakat használhatnak az egyes webhelyeken, és biztonságosan tárolhatják őket. Ezek a megoldások általában kétféle módon működnek:

Az egyik megoldás az automatikus jelszókezelés, amely lehetővé teszi a felhasználóknak, hogy egyszerre több jelszót is létrehozzanak és kezeljenek. Ezzel a megoldással a felhasználónak nem kell minden alkalommal új jelszót létrehoznia, ha elfelejt egyet. Emellett biztonságosabb is, mivel minden jelszót titkosítva tárolnak.

A másik megoldás pedig a manuális jelszókezelés, amely lehetővé teszi a felhasználónak, hogy saját maga hozza létre és kezelje az összes jelszavát. Ez az opció nagyobb szabadságot ad a felhasználónak, de ugyanúgy fontos figyelni arra, hogy ne felejtsen el bizonyos jelszavakat vagy ne adjon ki olyan információkat, amelyek segíthetnének másoknak hozzáférni az adataihoz.

- Jogosultságok kezelése:

A jogosultságok célja, hogy a felhasználók, hozzáférjenek bizonyos szolgáltatásokhoz / funkciókhoz. A jogosultságkezelés lehetővé teszi a rendszer adminisztrátorának, hogy korlátozza a felhasználók

hozzáférését bizonyos információkhoz és szolgáltatásokhoz. Az adminisztrátoroknak lehetőségük van arra is, hogy meghatározzanak bizonyos jogosultsági szinteket, amelyek meghatározzák, hogy milyen információkat és szolgáltatásokat érhetnek el a felhasználók. A webalapú rendszerek felhasználókezelési megoldásai segítenek a visszaélések megelőzésében is. Ez segít megelőzni az illetéktelen hozzáférést és visszaéléseket.

- Szerepkezelés:

A felhasználók szerepei lehetnek adminisztrátorok, ügyfelek, munkatársak vagy bármely más csoport. Ezek a jogosultságok meghatározzák, hogy az adott felhasználó mit tehet a

weboldalon. Például egy adminisztrátor szerep jogosult lehet arra, hogy hozzon létre és módosítson tartalmat, míg egy ügyfél csak olvashatja a tartalmat. A webalapú rendszerek felhasználókezelési megoldásai a biztonságot is szem előtt tartják. Amikor bejelentkezik a felhasználó, a rendszer automatikusan ellenőrzi az engedélyeket, ezzel megakadályozva, hogy illetéktelen személyek hozzáférjenek.

- Hozzáférés-vezérlés:

A webalapú rendszereken a hozzáférés-vezérlésnek köszönhetően, egyes felhasználók képesek lehetnek arra, hogy korlátozzák más felhasználók hozzáférését az adott rendszerhez. A hozzáférés-vezérlések lehetővé teszik egyes felhasználóknak, hogy létrehozzanak biztonsági profilokat, illetve jogosultságokat, amelyek meghatározzák, hogy mely funkciókat használhatnak egy adott rendszerben. Ezek a megoldások lehetővé teszik az adminisztrátorok számára is, hogy ellenőrizzék és kezeljék a felhasználói fiókokat. Ezen kívül segítséget nyújtanak abban is, hogy visszaigazolják a felhasználói hitelesítés helyességét a biztonságosságának fenntartásának érdekében.