Szegedi Tudományegyetem

Informatikai Intézet

SZAKDOLGOZAT

Urban Roland

2020

Szegedi Tudományegyetem
Informatikai Intézet

CRM rendszer megvalósítása Spring Boot keretrendszerrel

Szakdolgozat

Készítette:
Urban Roland
gazdaságinformatikus
hallgató

Témavezető:
Dr. Bodnár Péter
egyetemi tanár

Szeged
2020

FELADATKIÍRÁS

A feladat egy CRM rendszer megtervezése és implementálása. A webes alkalmazásban tárolt adatok egy tetszőlegesen választott adatbázisban legyenek elhelyezve. A projekt tetszőleges nyelven elkészíthető.

A rendszerrel szemben támasztott főbb követelmények az alábbiak:

- Képesnek kell lennie a hibajegyek feltöltésére, tárolására és annak megjelenítésére.
- Meg kell tudnia különböztetni a hibajegyeket státuszuk szerint.
- Lehessen regisztrálni, amivel személyre szabott adatok érhetőek el.
- A hibajegy prioritásának kezelése.
- A hiba továbbítása a megfelelő csoport számára.
- Keresés a hibák és az ügyfelek közt
- Statisztikai kimutatások készítése.
- A vezetői felületen az új felhasználók regisztrációjának elfogadása vagy elutasítása.
- Képes legyen kezelni a hibajegyekhez rendelt adatokat:
    - név
    - bejelentő
    - bejelentés típusa
    - a hiba prioritása
    - a bejelentés címe
    - leírás
    - dátum
- A hibabejegyzés életciklusának kezelése, nyomon követése.
- Támogassa a felhasználók kijelentkezését a rendszerből.

TARTALMI ÖSSZEFOGLALÓ

TARTALOMJEGYZÉK

1. BEVEZETÉS

Manapság az Internet mára egy világméretű hálózattá nőtte ki magát, amit több millió ember használ. Ahogy növekedett, úgy jelentek meg újabb szabványok, technológiák és ajánlások. Napjainkban, nem csak szükséges, hanem elengedhetetlen, a használat mind a hétköznapi felhasználóknak, mind a vállalatoknak is.

A web-es technológiák már régóta érdekelnek így amikor a szakdolgozati témaválasztásra került a sor, számomra egyértelmű volt, hogy olyat jelölök meg ami része ennek a nagy rendszernek. Mindig is fontos volt, hogy olyan technológiákat ismerjek meg, melyek napjainkban is helytállnak és a jövőben is helyt fognak. A Szegedi Tudományegyetem

folytatott tanulmányaim alatt megismerkedtem rengeteg ilyennel, de ezek közül kiemelném a Java-t, melyet a leggyakrabban használtam nem csak az egyes kurzusokhoz, hanem hobbi projektekhez is.

A dolgozat megírásának további pozitívuma lehet számomra, hogy nagyobb rálátásom lesz az alkalmazás-tervezésre, illetve a fejlesztésre, így bízom benne, hogy ezen ismeretek birtokában nagyobb lehetőségem lesz e területen elhelyezkedni

Szakdolgozatom témája egy CRM rendszer megtervezése és implementálása. A dolgozat első részében magáról a CRM rendszerről írok, majd a feladat megoldása során felhasznált technológiákat ismertetem és részletezem. Ezután a tervezési folyamatról írok hosszabban. Ezt követően az implementációs folyamat részletezése következik. Végül a szakdolgozatot egy összegzés zárja.

2. ÜGYFÉLKAPCSOLAT-MENEDZSMENT

2.1 A CRM általános jellemzői

Az ügyfélkapcsolat-menedzsment (Customer Relationship Management, CRM) tulajdonképpen új ügyfélkapcsolatok létrehozásával, ezek fenntartásával, valamint az ügyfelekről szóló információk felhalmozásával és megőrzésével foglalkozik.

Egy CRM rendszer segítségével a saját ügyfelekről egy információs adatbázist lehet létrehozni, amelybe a szervezethez tartozó minden felhasználó rögzíthet új értékesítési lehetőségeket, feladatokat. Ha az ügyfelekkel kapcsolatos minden információ egy helyen van, rendszerezetten tárolva, akkor a rendszert használók könnyebben, szervezettebben és hatékonyabban működhetnek együtt az ügyfelekkel és egymással is.

1. ábra
CRM rendszer összetétele
(Forrás [3])

A legtöbb CRM rendszer építőelemei:

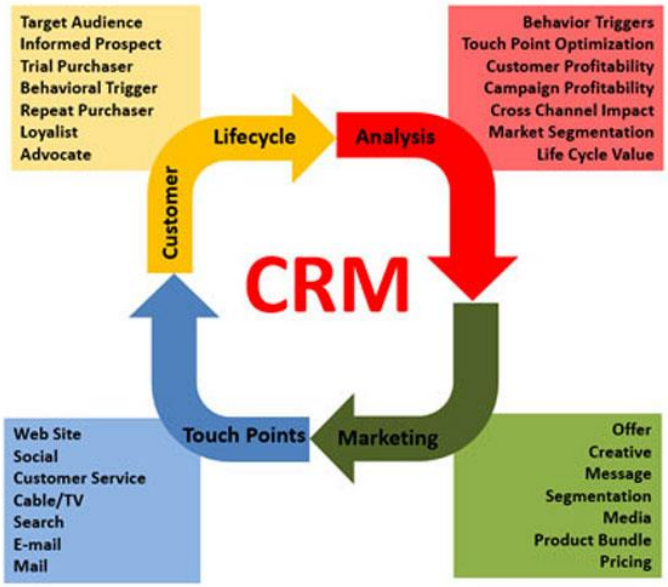
- **Ügyfelek**: lehetnek természetes személyek vagy kapcsolattartók abban az esetben, ha az ügyfél egy szervezetet jelent.
- **Adatlapok**: ezeken kerülnek tárolásra az egyes ügyfelekhez köthető értékesítési lehetőségek, segítségükkel követhető és mérhető az üzletkötés folyamata.
- **Teendők**: az ügyfelekkel történt kapcsolattartási események (e-mail küldés, telefonhívás stb.) feljegyzései, valamint a jövőbeni feladatok, emlékeztetők, amelyek az adatlapokon kerülnek rögzítésre.

Ezekből létrejön egy adathalmaz, ami lehetővé teszi többek között egy személyre való keresést, láthatók a hozzá kapcsolódó múltbeli értékesítések és jövőbeni lehetőségek, valamint az összes vele történt interakció.

Az alábbi területekre alkalmazható ez a rendszer:

- call centerek fejlesztése, átszervezése, telepítése, létesítése
- e-business
- értékesítési módszerek
- tréningek, gyakorlati támogatás
- ügyfél és piac szegmentáció
- értékesítési hatékonyság, stratégia, csatornák
- ügyfélelégedettség mérése

2.2 A CRM rendszer körforgása



2. ábra
CRM körfolyamata
(Forrás[4])

Mint már fentebb említettem a CRM alkalmazások központjában az ügyfelekről begyűjtött adatok állnak, hiszen ezek alapján tud a vállalkozás feléjük nyitni. Ezen adatok az ügyfelek általános adataira, termék vagy szolgáltatásokra, a velük megkötött megállapodásokra és az általuk előnyösnek vélt szállítási és egyéb szolgáltatásokra vonatkozhatnak. Ezek természetesen függenek a vállalkozás típusától, így egy banknak fontos lehet, hogy az ügyfél mennyit utazik külföldre, míg a szakácsnak nem biztos, hogy ez az információ releváns lenne. Az adatokat először be kell gyűjteni a klienstől, majd ezeket analizálni kell. Amennyiben ez a folyamat sikeresen megtörtént úgy a cég már könnyen el tudja dönteni, hogy az adott ügyfélnek mire van szüksége, így könnyebben bevonzhatja az ügyfelet, és elkezdheti a kapcsolatépítést. Ezután a kapcsolatot fenn kell tartania, amihez még több adatra van szüksége az ügyfélről. Ekkor újra elkezdődik az információszerzés, melyet ismét analizálás követ és az eljárás egy körforgásként működhet tovább.

3. FELHASZNÁLT TECHNOLÓGIÁK

A projekt elkészítésénél használt különböző komponensek és technológiák a következők:

A rendszert az Spring Tool Suite(STS)nevű nyílt forráskódú, platformfüggetlen szoftverkeretrendszer segítségével fejlesztettem, Java programozási nyelven.

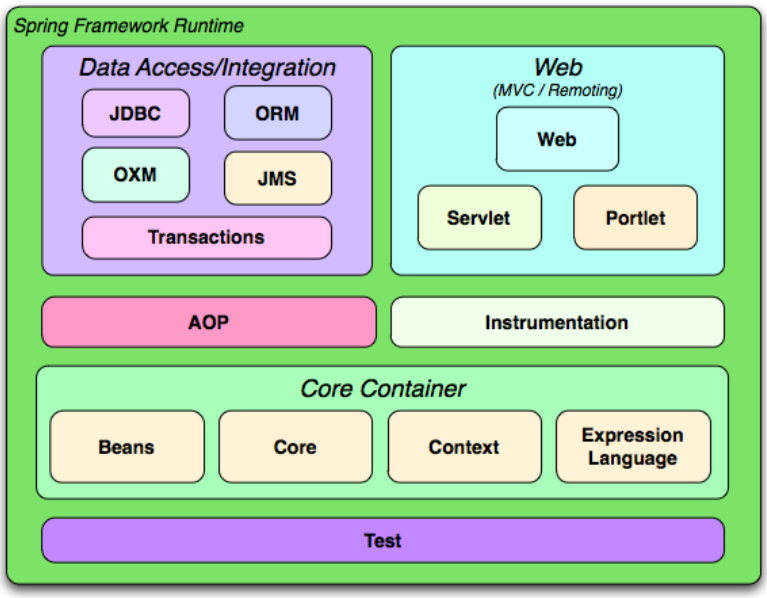
A projektet az Apache Maven plugin segítségével hoztam létre különböző modulokká felosztva és a Maven segítségével épül fel az alkalmazás.
A webalkalmazást a Spring Boot keretrendszer használatával fejlesztettem.
Adattárolásra PostgreSQL relációs adatbázist használtam.
A webalkalmazást Apache Tomcat szerveren fejlesztettem, teszteltem.
A webes felület a Bootstrap front-end keretrendszerrel, FontAwesome betűtípus és ikonkészlettel valósítottam meg.
A projekt hosztolásához a Herokut választottam, lényegében ez egy platform szolgáltatás (PaaS) webalkalmazásokhoz
A projektet készítése során az Git verziókövető rendszert használtam

3.1 Spring és Spring Boot

Ahhoz, hogy a teljes projektet egy nagy egészként tudjam kezelni, szükségszerű volt egy keretrendszerre használata. Ebbe a rendszerbe elhelyeztem magát az alkalmazást, így lehetőséget kaptam arra, hogy különböző szolgáltatásokat vegyek igénybe, amelyek megkönnyítették és felgyorsították számomra a fejlesztést.
Először a Spring-et mutatom be, eztután tovább haladva a Spring Bootot is, hiszen az alkalmazásomat ennek a keretrendszernek a használatával valósítottam meg.

A Spring egy nyílt forráskódú keretrendszer, amely az IOC (Inversion of Control, fordított vezérlés) tervezési mintára alapszik Java nyelven. A Spring gyakorlatilag egy IOC konténer, az IOC a Dependency Injection (függőség injektálás) támogatásával végzi el. Szoftverfejlesztés során az egyes elemek működése gyakran függ más elemek által nyújtott szolgáltatásoktól. Jellemzően tudniuk kell, hogy mely komponensekkel kommunikáljanak, hogy azok hol találhatóak, és hogy miként kommunikáljanak velük. Ha egy ilyen szolgáltatás valamilyen módon megváltozik, akkor azt valószínűleg elég sok helyen át kell írni. A szokásos módszer a kód struktúrálására az, hogy a szolgáltatások elérésének módját az alapvető logika részeként implementáljuk. Egy másik metódus, hogy elkerüljük az előzővel járó problémákat, hogy csak függőségeket deklarálunk a szükséges szolgáltatásokhoz, és valamilyen módon egy külső kódrészlet felel majd ezek megtalálásáért és inicializálásáért. Ez engedélyezi, hogy a szolgáltatások megváltozása esetén az ezeket használó kódrészleteket ne kelljen módosítani. Ezt a típusú módszert hívjuk Dependency Injection-nek.

A Spring egyetemes megoldásokat kínál a fejlesztés során felmerülő problémákra, a gyors és eredményes alkalmazásfejlesztésre, illetve rétegelt alkalmazások készítésére. A Spring rétegelt összetételű, ami azt jelenti, hogy a különböző fajtájú feladatok elvégzésére különböző modulok vannak. Ezek a modulok hat kategóriába sorolhatók (3. ábra). Ilyen például az adatbáziskezelés, webes szolgáltatások, tesztelési megoldások, amelyek mind külön modulokban találhatók.
Ennek a modularitásnak az a jelentősége, hogy nem kell az alkalmazást teljes mértékben a Spring keretrendszertől függővé tenni, nem kell hozzákapcsolni magunkat. Szabadon



3. ábra
Spring modulok csoportosítása
(Forrás[4])

dönthetünk arról, hogy mely modulokat használjuk, ami konkrétan kell, illetve, ha valamelyik modul nem megfelelő a számunkra, akkor egyszerűen egy másik keretrendszer könyvtárát használjuk a Spring által kínált megoldás helyett.

Tény az, hogy a Spring megkönnyíti a fejlesztés folyamatát a modulok megalkotásával, de maga a szoftver eltérő beállításait, mint például a Maven függőségek hozzáadását és az XML fájlok konfigurálását a fejlesztőnek kell kiviteleznie. Erre a problémára kínál megoldást a Spring Boot
Ez a keretrendszer a Spring család egyik tagja, amely olyan plusz funkciókkal rendelkezik a Springhez képest, amelyekkel még eredményesebbé, hatékonyabbá és könnyebé teszi a fejlesztést. Az a célja, hogy a szoftverfejlesztők ne a program konfigurálásával foglalkozzanak, hanem koncentráljanak a megírandó kódra, és hogy

azt az időt, amit a beállítások módosításával töltenének, azt az üzleti logikára és az implementálására fordíthassák .

3.2 Maven

Az Apache Maven egy csomagoló keretrendszer Java projektekhez. Egyik legfontosabb feladata, hogy automatizál folyamatokat. A Maven egyrészről leírja, hogy miként is épül fel a projekt, másrészről pedig meghatározza a projekt függőségeit(dependency) más moduloktól, vagy külső függvénykönyvtáraktól, amely csomagokat a build folyamat során automatikusan le is tölt. Mind ezen információk egy xml fájlban vannak eltárolva, melynek neve pom.xml. A külső package-ket dinamikusan tölti le egy, vagy több repository-ból, ez általában a központi Maven repository. A letöltött csomagokat a Maven helyi gyorsítótárban tárolja.
A már említett pom.xml file (pom = Project Object Model) tartalmazza az összes olyan adatot, ami elengedhetetlen egy program buildeléséhez. Ez többnyire magában foglalja a projekt nevét, tulajdonosát és a függőségek listáját. A kiterjedtebb projekteket általában szétszedik különálló modulokká, ahol minden modulhoz külön tartozik egy pom fájl. Ilyenkor annak a projektnek a pom.xml fájlja, ami magában foglalja a modulokat, lesz a gyökere az alkalmazásnak, ennek segítségével könnyedén lehet buildelni az összes modult. Az én projektemnél csak egy modult használtam.

3.3 PostgreSQL

A PostgreSQL egy nyílt-forráskódú relációs adatbázis-kezelő rendszer (RDBMS). A relációs adatbázis az adatokat táblákba rendezi, szerkezete: az adatok azonos összetételű sorokként kerülnek be egy táblába, ez a rekord. A felépítést, azaz a megőrzendő adatokat az oszlopok határozzák meg, ez pedig a mező. A tábla így egy egyedtípust reprezentál. Ha egy mezőt elsődleges kulcsnak jelölünk ki, akkor annak értéke egyértelműen meghatározza a rekordot, nem ismétlődhet. Ezt kiaknázva megadhatjuk ezt a mezőt egy másik táblában idegen kulcsnak jelölve, majd lekérdezéseknél egyesíthetjük a táblákat a kulcsértékek alapján. Minden relációs adatbázis-kezelő rendszer, így a PostgreSQL is az SQL (Structured Query Language, azaz struktúrált lekérdező nyelv) nyelvet használja az adatbázis létrehozására, módosítására és az adatok lekérdezésére. A PostgreSQL biztosít grafikus felületet az adatbázisok kezelésére. Én a

munkám során parancssort és a pgAdmin-t használtam, ami egy PostgreSQL adatbázisok kezelésére írt grafikus felületet biztosító eszköz.

3.4 GIT

A GIT [12] egy nyílt-forráskódú verziókövető rendszer (version control system, VCS). A verziókövető rendszerek kezelik a fájlokat és mappákat, valamint eltárolják a rajtuk megvalósított változtatásokat. Ez nagymértékben megkönnyíti a csapatban végzett fejlesztési munkákat, mivel nyomon követhetjük a saját, illetve a mások által végzett módosításokat, hiba esetén akármikor visszatölthetjük a rendszer egy korábbi állapotát. Bármilyen hálózaton üzemeltethetünk GIT szervert, ennek okán egymástól távol is eredményesen dolgozhatunk. Az, hogy másfajta fejlesztők egymástól távol képesek ugyanazt az adathalmazt kezelni és módosítani, jelentősen megkönnyíti az együttműködést. Mivel a munka verziókezelt, nem kell attól tartani, hogy valami súlyos hibát, vagy minőségromlást okoz változtatásunk a rendszerben, ez esetben csupán vissza kell töltenünk egy régebbi verziót.
Némely verziókövető rendszerek egyúttal szoftver konfigurációs rendszerek is (software configuration management system, SCM). Ezeket a rendszereket arra tervezték, hogy tudják kezelni a forráskódot. Efféle például, hogy felismerik és képesek kezelni a különböző programnyelveket, valamint rendelkeznek szoftver fejlesztést segítő beépített eszközökkel is. A GIT azonban nem tartozik ezek közé, hanem egy általános verziókövető rendszer, amellyel bármilyen adathalmazt kezelhetünk. A verziókövető rendszer központja az ún. repository, ami a központilag tárolt adathalmazt jelenti. Ez legtöbbször fájlrendszer formájában tárolja az adatokat, faszerkezetbe van rendezve a fájlok és mappák hierarchiája. Akármennyi kliens kapcsolódhat a repository-hoz, olvashatja és írhatja a fájlokat. Eddig nem túl sok mindenben tér el egy fájlszervertől, amiben különbözik az az, hogy a repository nyilvántartja az összes változtatást, emlékszik a fájlok minden verziójára. A verziókezelő rendszerek másik alapvető eleme a munkapéldány (working copy), ez a verziókezelt fájlok lokális másolata az egyes kliensek gépén. A munkapéldányt a kliens először letölti a központi repository-ból, majd azon végzi el a változtatásait, fejlesztéseit, ezután hozzáadja a lokális tárolóhoz majd végül visszatölti azt a központi-ba, amennyiben nem történt olyan változás, ami ütközik egy másik kliens változtatásával. Ha az utóbbi megtörténik akkor először fel kell oldani a konfliktusokat, majd ezután válik lehetővé a feltöltés.

Ennek megoldásában is segítenek az GIT eszközök. Szoftverfejlesztésnél legtöbbször van egy törzs (trunk), ami a szoftver fő verzióit tartalmazza, továbbá vannak a másmilyen ágak (branch), amelyeken fejlesztik a módosításokat, új funkciókat hoznak létre és csak utána vezetik ezeket át a fő vonalra.

4.SPECIFIKÁCIÓ

A hibajegyek elbírálási folyamata során a rendszerben a vele kapcsolatba kerülő felhasználók különböző szerepkörökben különböző feladatokat látnak el. Ezek a szerepkörök határozzák meg az alkalmazás funkcionalitását. Most az egyes szerepkörök részletezése következik:

Vezető –

Ügyfélszolgálat –

Fejlesztő –

Tesztelő –

Hardverszerelő –

	Regisztrálás	Bejelentkezés	Új hiba bejegyzés rögzítése	Hibabejegyzés továbbítása	Hibabejegyzés lezárása	Statisztikák kimutatások	A felhasználók regisztrációjának jóváhagyása	Keresés az ügyfelek között	Keresés a hibajegyek között	Kijelentkezés
Vezetők	x	x	x	x	x	x	x	x	x	x
Ügyfélszolgálat	x	x	x	x	x			x	x	x

Fejlesztő	x	x		x	x		x		x	x
Tesztelők	x	x			x				x	x
Hardverszerelők	x	x			x				x	x

4. ábra
Szerep-funkció mátrix

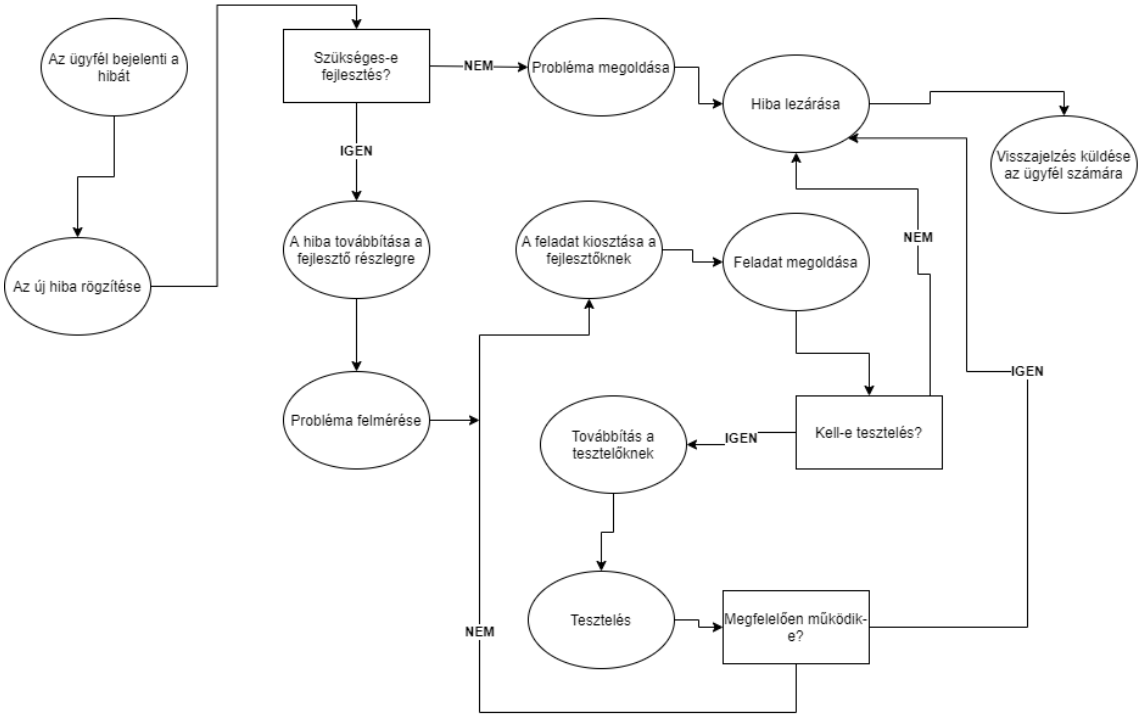
Egy hibajegy lehetséges állapotai a következők:

- Nyitott
- Zárt
- Folyamatban
- Felfüggesztett

A továbbított hibabejegyzések a következő sorrendben lesznek kezelve a továbbiakban:

- Kritikus hiba
- Gyorsfelmérés
- Extra fontos
- Nagyon fontos
- Fontos
- Normál
- Ráér

Az 5. ábra szemlélteti a hiba életciklusát, segítségével könnyebben megérthető, hogy a rendszer miként is kezeli a hibajegyeket.



5. ábra
A hiba életciklusa

5.TERVEZÉS

5.1 Modulok

6. IMPLEMENTÁCIÓ

A jelenlegi rendszer analízise

A jelenleg használt rendszert egy kisvállalkozásnál használták. A célja az volt, hogy az ügyfelek esetleges hibáit begyűjtse, tárolja és megoldja ezeket. A jelenlegi rendszer egyik legnagyobb hibája, hogy teljesen papíralapú. E tulajdonsága nem teszi lehetővé a különböző hibák gyors és hatékony megoldását. Illetve az egyre növekő adatok dinamikus kezelése igen nehézkessé vált. A vállalkozás belátta, hogy hosszú távon ez a rendszer nem kifizetődő így beruháztak egy saját CRM rendszerbe. Ez okból kérték, hogy tervezzek egy rendszert, ami kielégíti az igényeiket. Kisvállalkozás révént külön kérésük volt, hogy az egyszerű felhasználói felülettel mellett az új felhasználók regisztrációjának aktiválását a vezetők hagyják jóvá. A jelenlegi rendszer működése az alábbi folyamatokra bontható.

Hibák bevitele és feldolgozása

Az az ügyfél, aki valamilyen hardveres vagy szoftveres hibát észlelt felhívta a vállalkozás ügyfélszolgálatát, az ügyintézőnek elmondta a problémát. Ez rögzítésre került papír formájában. Amennyiben az ügyintéző nem tudott segíteni a megoldásban abban az esetben tovább küldte az illetékeseknek. Ők elemezték a hibát majd visszacsatolást küldtek az ügyintézőnek, aki értesítette az ügyfelet

Irodalomjegyzék

[1] https://www.soulware.hu/szakmai-reszletek/a-crm-reszletesebben/?gclid=Cj0KCQjwka_1BRCPARIsAMlUmErp3rORyrxHa-gWCO4X_QDosv66Lzhs_bacdrquNuRyX57OUhmBs4aAsMcEALw_wcB

[2] https://hu.wikipedia.org/wiki/CRM

[3] https://www.perfectviewcrm.com/what-is-crm/

[4] https://www.resonantanalytics.com/services/customer-relationship-management/

[5]https://spring.io/projects/spring-boot

[6] https://docs.spring.io/spring/docs/3.0.0.M3/reference/html/ch01s02.html

[7] https://hu.wikipedia.org/wiki/A_f%C3%BCgg%C5%91s%C3%A9g_befecskendez%C3%A9se

[8] https://hu.wikipedia.org/wiki/Apache_Maven

[9] http://maven.apache.org/index.html

[10] https://hu.wikipedia.org/wiki/PostgreSQL

[11] https://www.postgresql.org/

[12] https://hu.wikipedia.org/wiki/Git

[13] https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control

Nyilatkozat

Alulírott ……..…… szakos hallgató, kijelentem, hogy a dolgozatomat a Szegedi Tudományegyetem, Informatikai Intézet …….. Tanszékén készítettem, …….……. diploma megszerzése érdekében.

Kijelentem, hogy a dolgozatot más szakon korábban nem védtem meg, saját munkám eredménye, és csak a hivatkozott forrásokat (szakirodalom, eszközök, stb.) használtam fel.

Tudomásul veszem, hogy szakdolgozatomat / diplomamunkámat a Szegedi Tudományegyetem Informatikai Intézet könyvtárában, a helyben olvasható könyvek között helyezik el.

Dátum

Aláírás