

**Szegedi Tudományegyetem
Informatikai Intézet**

SZAKDOLGOZAT

Fehér Zsolt

2021

Szegedi Tudományegyetem
Informatikai Intézet

Számítógépes Optimalizálás Tanszék

**Kis- és Középvállalati ügyfélkapcsolat kezelő
rendszer megtervezése és megvalósítása**

Szakdolgozat

Készítette:

Fehér Zsolt

Gazdaságinformatika

szakos hallgató

Témavezető:

Csicsman József

Tanszékvezető:

Bánhelyi Balázs

Szeged
2021

Feladatkiírás

A hallgató feladata, hogy megtervezzen egy jól strukturált CRM rendszert, amelyet funkcionalitásában eleget tesz egy ügyfélkapcsolat-kezelő szoftver követelményeinek. Ezek alapján meg kell valósítania egy működő, bemutatásra alkalmas rendszert, a választott programnyelv, illetve keretrendszer segítségével. A feladat megoldásának tükröznie kell a tanulmányai során elsajátított informatikai ismereteket.

A munka elején ismerkedjen meg a kiválasztott technológiák és módszerek nyújtotta lehetőségekkel és ezek használatát készség szinten sajátítsa el a valódi fejlesztés megkezdése előtt. Tervezze meg az adatstruktúrákat és a rendszer alapvető felépítését.

Specifikálja a program funkcióit, szereplőit és részeit. A logikailag összetartozó részeket csoportosítsa és ezek szerint készítse el az alkalmazást. Az ügyviteli rendszer típusából eredően meg kell valósítania az ügyfelek adatainak kezelését és a hozzájuk tartozó értékesítések követését.

Az alkalmazás futtatható legyen, illetve tegye elérhetővé akár lokálisan, akár online telepített formában. Továbbá adja meg az ezekhez szükséges útmutatásokat és telepítési információkat.

Dokumentálja a fejlesztési folyamatot és lehetőség szerint használjon verziókövetést. A munkáját és az elért eredményeket egy szakdolgozat keretében foglalja össze és mutassa be.

Tartalmi összefoglaló

- ***A téma megnevezése:***

Témám a kis- és középvállalati ügyfélkapcsolat kezelő rendszer megtervezése és megvalósítása, amelyet az ügyviteli szakirodalom CRM rendszerekként is ismer. A 2008-as válság óta kiemelt fontosságú szerepet töltenek be a gazdaságban és ez a koronavírus járvány alatt sincs másképp.

- ***A megadott feladat megfogalmazása:***

A cél egy olyan rendszer megalkotása, amely képes az értékesítési folyamatok követésére és megjelenítésére, megvalósítja a felhasználókezelést, és képes a reaktív adatműveletekre, valamint ezek eredményeinek összegzésére, illetve csoportosítására.

- ***A megoldási mód:***

A rendszert, a TypeScript alapú keretrendszer, az Angular segítségével valósítom meg, amely a Google által fejlesztett Firestore felhőalapú adatbázissal kommunikál. Az applikáció elkészítése modulokra bontható, amelyek jelen esetben a webes felület oldalaként értelmezhetőek. Az egyes modulok újra felhasználhatóak, ezáltal megkönnyítve a program bővíthetőségét a későbbiekben.

- ***Alkalmazott eszközök, módszerek:***

A fejlesztés során két nagyobb UI komponens könyvtár kerül felhasználásra, az Angular saját fejlesztésű Angular Materials könyvtára, illetve a PrimeTek által fejlesztett PrimeNG. Ezen kívül a fejlesztési folyamat szakaszai a program GitHub oldalán megtekinthetőek.

- ***Elért eredmények:***

A fejlesztés végeredménye egy olyan felhasználóbarát, felhőalapú webalkalmazás, amely nagyban hozzájárulhat egy értékesítési szakcsoport munkájához. Nem csak a hasznos információk megjelenítésére, de a teljes értékesítési ügymenet követésére is alkalmas.

- ***Kulcsszavak:***

CRM, értékesítés, felhőalapú, webalkalmazás, ügyfélkapcsolat-kezelő, angular

Tartalomjegyzék

Feladatkiírás	2
Tartalmi összefoglaló	3
Tartalomjegyzék	4
BEVEZETÉS	6
1. AZ ÜGYVITEL ÉS AZ INFORMATIKA	7
1.1. Informatika a gazdasági folyamatokban	7
1.2. Az értékesítés folyamatai	8
1.3. Az ügyfélkapcsolat-kezelő (CRM) rendszerek	9
2. KÖVETELMÉNYSPECIFIKÁCIÓ	11
3. A FOX-CRM FELÉPÍTÉSE ÉS MŰKÖDÉSE	12
3.1. Technológiák és módszerek.....	13
3.1.1. Angular.....	13
3.1.2. Firebase	13
3.1.3. PrimeNG és Angular Materials.....	14
3.1.4. Fejlesztői környezet	14
3.2. Adatbázis modellek	15
3.3. Service-ek	17
3.4. Autentikációs modul.....	18
3.4.1. Regisztrációs oldal	18
3.4.2. Bejelentkezés oldal.....	19
3.5. Profil modul.....	19
3.6. Kezdőlap.....	20
3.7. Értékesítés-áttekintő modul	22
3.8. Értékesítés modul.....	23
3.8.1. Értékesítés oldala	24
3.8.2. A cég- és kapcsolattartó oldala	26
3.9. Termékek oldal.....	27
3.10. Statisztikai oldal	29

3.11. Egyéb elemek.....	30
3.11.1. A menüsáv	30
3.11.2. UI értesítések	30
 4. A FOX-CRM FUTTATÁSA.....	31
4.1. Lokális futtatás.....	31
4.2. Firebase Hosting.....	31
 5. A FOX-CRM BEMUTATÁSA	32
5.1. Autentikáció	32
5.2. Profilok kezelése.....	33
5.3. Értékesítések listája.....	33
5.4. Értékesítéskövetés	34
5.5. Termékek.....	36
5.6. Statisztikák	36
 6. BŐVÍTÉSI LEHETŐSÉGEK	38
 7. ÖSSZEFOGLALÓ	39
 Irodalomjegyzék	41
 Nyilatkozat	42
 Köszönetnyilvánítás	43
 Elektronikus melléklet	44

BEVEZETÉS

Mi egy vállalat legfontosabb bevételi forrása? Mire támaszkodik vagy ki biztosítja számára, hogy képes legyen túlélni a piac viszontagságait? A válasz: az ügyfélköre.

Minden stabil cég mögött áll egy hasonlóan lojális -esetenként pedig változékony- ügyfélkör. Ez lehet monopol ügyfél, például az állam, vagy eklektikus, mint egy független szoftverfejlesztő cég ügyfélköre. Ez utóbbi vállalatípus egy összetett problémával találkozhat szemben magát nap mint nap, ami nem más, mint az ügyfélinformációk, értékesítések és az ügyfélkapcsolatok látszólagos, de egyáltalán nem effektív menedzselése, egy megfelelő rendszer háttértámogatása nélkül.

Kis- és középvállalkozásoknál, ahol a szerepkörök nem modularizáltak, vagyis az alkalmazottak nem egyetlen szerepet töltenek be, hanem többet (például: egy időben értékesítés, be- és kivitel, termékmenedzsment), ott tipikus példa, hogy az ügyfél-információval csak bizonyos emberek rendelkeznek. Ideálisabb esetben ezek az információk le vannak írva fizikailag egy dokumentumban, viszont egy ennél rosszabb szituáció, ha az információ csak gondolati síkon létezik.

Ezek a módszerek lényegesen csökkentik a hatékony információáramlást a vállalaton belül, nem beszélve arról, hogy az ilyen tapasztalt értékesítő elvesztése a vállalat és az ügyfél kapcsolatának szempontjából felér egy visszaállítással.

Hogy ezeket a kockázatok elkerüljük, esetleg mérsékeljük, centralizálnunk kell az értékesítéseket és az ügyfélkapcsolatokat egy mindenki által követhető és könnyen kezelhető rendszerben. Ezeket a rendszereket nevezzük **Ügyfélkapcsolat-kezelő** rendszereknek.

1. AZ ÜGYVITEL ÉS AZ INFORMATIKA

1.1. Informatika a gazdasági folyamatokban

Az informatikai megoldások szinte mindegyik gazdasági szektorban jelen vannak, legyen szó akár pénzügyi vagy akár marketing szektorról. A technológiai fejlődésnek köszönhetően ez egy természetes integrálódás, hisz mindezek célja a műveletek gyorsabb, effektívebb és sikeresebb végrehajtása.

Az elektronikus kereskedelem, továbbiakban e-kereskedelem, az egyik legkorábbi ezek közül, illetve ez a szektor változott a legtöbbet. Ebbe a kategóriába tartoznak, a személyek mellett, a vállalatok közötti értékesítés és egyéb kereskedelmi folyamatok informatikai megvalósításai. Elég csak a webáruházakra gondolnunk, amely jól szemlélteti az érintésmentes, tisztán informatikai áruvásárlást. Nem beszélve azokról az információkról, amelyeket akár csak egy ilyen vásárlás is generálhat az értékesítő cég számára.

Egy másik nagyon fontos informatikai megoldás, vállalati szemszögből és aminek szakdolgozatom is a része, az ERP rendszerek. Angolul **Enterprise Resource Planning Systems**, viszont magyarul Vállalatirányítási rendszerkekként hivatkozunk rájuk. Ez egy átfogó elnevezés, lényegében egy ilyen rendszernek nevezzük azoknak az alrendszerek egymásra támaszkodó kollektív csoportjait, amelyek egy vállalat működéséhez szükséges folyamatok szinte mindegyikét lefedik. Az alrendszerek többek között a Termékmenedzsment, a HR vagy a **CRM** rendszerek. Mindegyik egy bizonyos szektort fed le egy cég gazdasági életében. Ha absztraktabbul szeretnénk szemlélni ezeket az alrendszereket, akkor csoportosíthatjuk őket négy modulra:

- **Ügyvitel** (például: CRM)
- **E-kereskedelem** (például: Beszerzéstervező rendszer, avagy SRM)
- **Vállalat erőforráskezelés** (például: HR)
- **Termelési rendszerek** (például: Folyamatkövető rendszer)

Közös jellemzőjük, hogy integrált, valós idejű műveletekre képes rendszerek, továbbá kollektív adatbázissal rendelkeznek, illetve kialakításuk és felhasználói felületük is konzisztens.

Természetesen egy ilyen összetett struktúra megvalósítása nem egy apró feladat és van, hogy a megrendelőnek nincs is szüksége ilyen robosztus rendszerre, hanem csak egy önálló, jól elkülöníthető részére. Ezeket nevezzük célszoftvereknek.

1.2. Az értékesítés folyamatai

Amikor valaki értékesítésre gondol, talán a leggyakoribb koncepció, ami elé tárul, egy egyszerű porszívóügynök, vagy autókereskedő. Viszont létezik egy ennél jóval egyszerűbb fajtája is, még pedig a tömeges értékesítés. Legyen az háztartási eszköz vagy élelmiszer, mind-mind értékesítésnek minősülnek. Habár eleget tesz az értékesítés definíciójának, mégsem ez a legelső, ami eszünkbe jut.

Az értékesítés, mint folyamat egy több szakaszos tevékenységsorozat, amely fajtájától függően állhat több, illetve kevesebb lépésből. Egy tej vásárlása értékesítési szempontból nem egy hosszú lépéssorozat. Viszont egy lakásvásárlás ingatlanirodán keresztül már egy egészen hosszú folyamatot eredményez. Talán egyértelmű, de a különbség nem csak a termék vagy szolgáltatás értékbeli különbségében nyilvánul meg, hanem az igény felmérésében és ezen információ kezelésében is.

Egy személyre szabott értékesítés ügyfélközpontú, standardizált mégis rugalmas, illetve követhető lépések egymást követő sora, amelynek eredménye ideális esetben megjósolható sémát követ. Egyszerűsített lépései:

- I. Kutatás
- II. Előkészítés
- III. Felmérés
- IV. Prezentálás
- V. Lezárás

A folyamat első felében maga az ügyfél keresése, piackutatás és információ gyűjtés zajlik. Ebben a szakaszban ismerjük meg az ügyfelet, kategorizáljuk és felkészülünk a vele való kommunikációra. Ezután demonstráljuk az általunk értékesíteni kívánt terméket és reagálunk ennek megfelelően. Végül pedig a lezárás következik, ahol megtörténik a szerződéskötés vagy épp a sikertelen értékesítés okainak felkutatása.

Ezen szakaszok határai ügyfelenként és értékesítőnként elmosódhatnak ezért feloszthatjuk őket három absztraktabb csoportra: Felmérés, Bemutató (Folyamatban van), Lezárás. Mindegyik csoport több, kisebb lépésből is állhat, viszont az egyszerűbb és egységesített folyamatábrázolás miatt ezt az absztrakciót valósítjuk meg.

1.3. Az ügyfélkapcsolat-kezelő (CRM) rendszerek

A bevezetőben már felvezettem az ügyfélkapcsolat-kezelő rendszerek szükségességét, viszont érdemes kitérni, hogy mit is értünk CRM (Customer Relationship Management) rendszer alatt.

Olyan rendszereknek nevezzük ezeket, amelyek egy vállalat ügyfeleinek adatait tárolja, feldolgozza és elősegíti a velük való kapcsolattartás. Az ügyfelek alatt több különböző partnerscsoportot értünk: múltbéli ügyfelek (nem aktív), jelenlegi ügyfelek (aktív) vagy potenciális ügyfelek. Világosan látható, hogy a rendszer kezeli és tárolja a teljes ügyfélkörünket, ezáltal egy jól átlátható struktúrát biztosítva a felhasználónak.

Természetesen ezeket egy komplexebb adatbázis is képes lenne elvégezni. Viszont a CRM rendszerek igazi jelentősége az értékesítések támogatásában rejlik. Elősegítik az értékesítési folyamatok követését, értékesítési stratégiák tervezését és ügyfelek kategorizálását. Statisztikákkal támogatják az értékesítők munkáját, illetve információkat szolgáltatnak a marketing vagy a vezetői szektornak.

De milyen adatok vagy információk is ezek? Egy ügyfél rengetek adattal rendelkezik, nem csupán **technikai**, hanem **egyéni**, **habitus jellegű** adattal is.

Előbbit feloszthatjuk Kapcsolattartáshoz szükséges adatokra és Céginformációkra. Ilyenek például a kapcsolattartó személy telefonszáma vagy a cég adószáma. Az egyéni és habitus jellegű adatokhoz soroljuk azokat az adatokat, amelyek az ügyfél valamely állapotához vagy szokásához köthető. Előbbi információ lehet például végzettség vagy családi állapot, míg az utóbbira egy remek példa, hogy mikortól meddig elérhető az ügyfél telefonon. Ezek mindegyike hozzájárul az értékesítés sikerességéhez, hisz még a legjelentéktelenebbnek tűnő adat is óriási különbséget jelenthet egy folyamat eredményében.

Miután tisztáztuk egy CRM rendszer adatrendszerének elemi felépítését, térjünk ki a rendszer típusaira. Három fajtáját különböztetünk meg: **Műveleti**, **Elemző** és **Kollaboratív** CRM-ek.

Műveleti vagy operatív ügyfélkapcsolat-kezelő rendszerekről akkor beszélhetünk, amikor a legfontosabb tevékenysége közé az értékesítés, marketing és ügyfélkezelés folyamatainak elősegítése tartozik. Értelemszerűen ez a legelterjedtebb fajtája a három közül.

Az elemző vagy analitikus ügyfélkapcsolatkezelő rendszerek, olyan funkciókra összpontosítanak, amelyek a meglévő adatokat elemzik, főként a marketing és az értékesítési szektort támogató információk értelmezése céljából. Az így kinyert adatokat pedig ideális esetben visszaküldik a műveleti CRM rendszerekbe.

Kollaboratív vagy együttműködő CRM, az a rendszer, amely nem csak az értékesítők számára osztja meg az információkat, hanem a vállalaton belül működő több szektornak is. Ezen kívül megvalósítják a valódi kommunikációt az értékesítő és az ügyfél között. Ilyenek például a „call center” alkalmazások is.

Megemlítendő, hogy ezek esetenként összeforrhatnak, a megrendelő igényétől függően. A továbbiakban egy ilyen rendszerrel foglalkozunk majd, ami főként egy Operatív CRM, viszont rendelkezik analitikus jellemzőkkel.

2. KÖVETELMÉNYSPECIFIKÁCIÓ

A cél egy olyan cloud-based szoftver létrehozása, amely támogatja az értékesítők és elemzők munkáját és kielégíti egy ilyen jellegű rendszer felé támasztott alapvető elvárásait a következő funkciókkal:

- Felhasználókezelés
- Profil módosítása
- Értékesítések létrehozása és szerkesztése
- Értékesítések nyomon követése
- Értékesítési előzmények megjelenítése
- Teendők hozzáadása, szerkesztése és törlése
- Céginformációk létrehozása és módosítása
- Termékinformációk listázása, létrehozása, szerkesztése és törlése
- Statisztikai elemzések megjelenítése

A fenti funkciók logikailag csoportra oszthatók, amelyek ekvivalensek tekinthetők a megvalósítandó oldalak egy részével. Ezek a csoportok az Értékesítések, a Felhasználókezelő és autentikációk, a Céginformációk, Termékinformációk és végül a Statisztikák.

A megjelenítési követelmények a letisztult mégis esztétikus kinézetet és a lényegi információmegjelenítést foglalják magukba. A vizualizáció egyszerűsítésének fő oka, hogy a felhasználót ne érje túl sok információs inger a rendszer használata során.

3. A FOX-CRM FELÉPÍTÉSE ÉS MŰKÖDÉSE



3.1.: ábra: FOX-CRM használati eset diagramja [Saját forrás]

Az általam megtervezett rendszer a 3.1.-es ábrán látható használati eset (use case) diagram szerinti funkciókat és kapcsolatokat fogja megvalósítani. Alapvetően egy felhasználói szerepkört indokol jelenleg a rendszerterv, ami nem kizárt, hogy a jövőben esetleg egyéb felhasználókkal is kiegészül. A külső szolgáltatások mindegyike a Firebase-hez tartozik, ezek pedig az autentikációs, saját adatbázis és a saját tárhely service-k. A rendszerben lévő folyamatok ez a három szolgáltatás szerint csoportosulnak.

Az applikáció megtervezésében és elkészítésében egyedül én vettem részt, mint fejlesztő. A rendszer neve FOX-CRM lett, a továbbiakban pedig így hivatkozom majd rá.

3.1. Technológiák és módszerek

3.1.1. Angular

A szakdolgozatom során az Angular keretrendszerrel foglalkoztam, amely lefedi azt a technikai hátteret, amely szükséges a szakdolgozatom elkészítéséhez. Ideértve a teljesen alap tudást, mint egy egyszerű weboldal készítése, illetve összetettebb elemeket és funkcionalitásokat, mint például a felhő alapú adatbázis szolgáltatások vagy az RxJs.

Alapvetően az Angular egy TypeScript-re épülő webfejlesztési keretrendszer, amelyet a Google és független fejlesztők közösen fejlesztenek, és amely iszonyatosan népszerű lett az elmúlt években. A struktúrája egy modulokból álló, komponens alapú web alkalmazások elkészítését teszi lehetővé. Rendkívül progresszív technológia és elképesztő technikai mélységű alkalmazások készíthetők vele. Az Angular közösségnek hála pedig rengeteg könyvtár érhető el különféle feladatok vagy problémák kiküszöbölésére.

Egy nagyon könnyen tanulható, kliens-oldali, front-end keretrendszerről beszélünk, amely támogatja az MVC megközelítést és Dependency injection-t (DI). Rengeteg ismert weboldal van, amelyet Angularban írtak, mint például a Gmail, a Forbes, illetve a PayPal.

3.1.2. Firebase

A Google Firebase egy mobil és web applikációk fejlesztését támogató platform. Egy nagyon összetett eszköztár, amiről akár egy egész szakdolgozat is szólhatna, mivel olyan mélységekkel rendelkezik.

Tömören összefoglalva olyan eszközöket biztosít a számunkra, amelyek lefedik azokat a szolgáltatásokat, melyekre a legtöbb fejlesztőnek szüksége van, pontosabban biztosít számunkra elérést: Real-Time adatbázishoz, Autentikációs szolgáltatáshoz (amellyel akár Facebook vagy Twitter fiókkal is bejelentkeztethetünk felhasználókat), Tárhely szolgáltatáshoz, Értesítés menedzserhez és még sok hasonlóan hasznos funkcióhoz.

Lényegében a Firebase biztosítja számunkra a back-end komponenseket, közvetlen módon, mindenféle köztes szoftver bevonása nélkül.

Ha komolyabban szeretnénk kihasználni ezt a támogatást, akkor lehetőségünk van az applikációnk különböző analitikai monitorozására, például mekkora forgalma van, milyen gyakran omlik össze, vagy hogy milyen a teljesítménye. Ezen felül egy kiszolgáló szolgáltatást is kapunk, melynek segítségével a Firebase szerverén futtathatjuk az applikációnkat.

Szakdolgozatom szempontjából kiemelendő szolgáltatás az úgynevezett Firestore felhő alapú adatbázis és a már említett tárhely szolgáltatás. A Cloud Firestore sok tekintetben effektív adatbázis funkció, nem csak hogy valós időben biztosít felhő alapú szinkronizációt, de lehetővé teszi a reszponzív applikációk megvalósítását, függetlenül az internetünk minőségétől. Hierarchikus adatstruktúrája lehetővé teszi, a komplexebb objektumok tárolását, kezelését és feldolgozását. Alapvetően kollekciókból épül fel, amelyek dokumentumokat tartalmaznak. Ezek egy entitásnak feleltethetőek meg, amelyeken megvalósíthatók a CRUD műveletek. A tárhely szolgáltatás, ahogy a nevéből is lesűrhető, egy szabadon használható fájl tárhelyt biztosít az rendszerünknek.

3.1.3. PrimeNG és Angular Materials

Egy applikáció fejlesztésekor előnyt jelent úgynevezett külső UI könyvtárak használata. Ezek segítségével gyorsabban és sok esetben szebb megjelenéssel készíthetők tartalmasabb és felhasználóbarát felületek. Két ilyen könyvtárat használok a munkám során: A PrimeNG-t és az Angular Materials-t.

A PrimeNG egy kifejezetten gazdag UI elemek könyvtára. Az mellett, hogy teljesen ingyenes és „open source”, még látványos is. Mind megjelenésében mind használatában rendkívül modern stílus érzését nyújtja a felhasználónak. A könyvtárat a török PrimeTek fejleszti, amely felelős a JSF (JavaServer Faces) alapú applikációk egyik közkedvelt UI könyvtárának, a PrimeFaces-nek, a megalkotásáért. Implementálása is végtelenül egyszerű, olyan, mint bármely angular könyvtár telepítése, amelyet npm csomagkezelővel futtathatunk.

Továbbá megemlítenő az Angular saját fejlesztésű, hasonló jellegű könyvtára az Angular Materials. Szintén egy nagyon jó minőségű komponenscsomag, amely az egyik legmegbízhatóbb opció. Nem olyan széleskörű komponens-fajtákkal rendelkezik, mint az előbbi, viszont amelyekkel igen, azok valóban szükséges UI elemek.

3.1.4. Fejlesztői környezet

A fejlesztéshez a Microsoft által fejlesztett, ingyenes kódszerkesztőt használom, a Visual Studio Code-t. Ez egy teljesértékű kódszerkesztő, amely támogatja az Angular programozást több aspektusból is. Termináljából elérhető a telepített Angular CLI (Command Line Interface), amivel lokálisan tudjuk futtatni a webalkalmazásunkat.

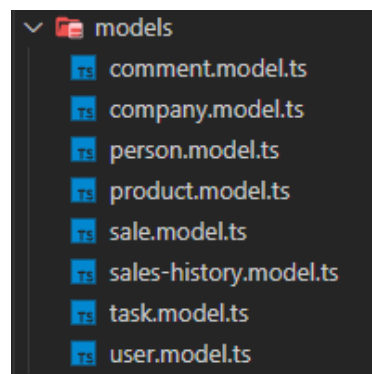
Mivel szüksége van egy futtatási környezetre (Server Runtime Environment), ezért szükséges egy Node.js környezet telepítése. Az alábbi verziók mellett biztosított a program integritás:

- Angular és Angular CLI: **v10.2**
- Node.js: **v14.15**
- PrimeNG: **v11.0**
- Firebase: **v8.4.2**
- Chart.js: **v2.9.4**

3.2. Adatbázis modellek

Az Angular szempontjából, a hagyományos adatmodellek interface-ként jelennek meg a programban. Ezek biztosítják az adat konzisztens alakját, vagyis egy olyan formátumra kötelezik az objektumot, amely rendelkezik a megfelelő tulajdonságokkal és eleget tesz az általunk definiált kritériumunknak. Így garantáljuk, hogy az adatbázisba bekerült elemek attribútumai kellő számban és kellő típusként vannak jelen.

Jelenleg az alkalmazásban nyolc ilyen adatmodell leírás található (3.2.1. ábra), amelyeket aktívan használ a rendszer. Szinte mindegyikük kapcsolatban áll egy szomszédos adatmodellel külső kulcsok által. Például egy Vállalat modellje hivatkozik egy Személy modellre a külső kulcsa segítségével ezáltal testesítve meg a Vállalati-Kapcsolattartói viszonyt. Így egy vállalat indirekt módon eltárolja a kapcsolattartó személyét is, egy hivatkozáson keresztül, ami jelen esetben a személy ID-ja.



3.2.1. Az alkalmazás modelljei [Saját forrás]


```

3   type TimeStamp = Firebase.firestore.Timestamp;
4
5   export interface IComment{
6       id?: string;
7       text: string;
8       createdBy: string;
9       createdAt: TimeStamp;
10      isEdited: boolean;
11      updatedAt: TimeStamp;
12      updatedBy: string;
13      userId: string;
14  }

```

```

export interface ISale{
    id?: string;
    saleId: string;
    status: string;
    companyId: string;
    responsibleId: string;
    expectedDate: TimeStamp;
    products: string[];
    createdAt: TimeStamp;
    createdBy: string;
    closingDate: TimeStamp;
    expectedIncome: number;
    closingIncome: number;
    concerns: string;
    closingReason: string;
    customerType: string;
    surveyInfo: string;
    progressInfo: string;
    comeFrom: string;
}

```

3.2.2. ábra: Egy komment és egy Értékesítés modellje [Saját forrás]

Az ilyen jellegű adatleírások ismerősek lehetnek az MVC struktúrában jártas személyek számára, ugyanis itt is, hasonlóan a struktúra Modell mintájára, szabályokat fogalmazunk meg, hogy milyen adatokkal rendelkezzen az adott entitás. Ahogy a példa ábrán (3.2.2. ábra) is látszik, az adattag nevét az elvárt típusát is definiáljuk az interfaceben.

Szembe tűnhet, hogy a fenti ábrán helyet kapott egy nem elemi adattípus is. Az úgynevezett Firebase TimeStamp a Google saját „időbélyegje”, amely számsorozatként tárolja az adott dátumot és időt. Az adatbázis egyszerűbb kezelhetősége miatt vezettem be több helyen is.

Ezekkel az interfacekkel a későbbiekben objektum példányokat tudunk specializálni, tehát egy fiktív keretet szabni a tulajdonságainak. Gyakran elő fog fordulni különböző adatbázisműveletek végrehajtásakor vagy az eredményük tárolásakor.

Kétségtávol a legterjedelmesebb modell az Értékesítési (ISale) interface (3.2.2. ábra), amely egy értékesítés összes információját és hivatkozásait tárolja. Nem meglepő, mivel lényegében ez az elem a CRM rendszerünk alapköve.

3.3. Service-ek

Ha már az alapköveknél tartunk, térjünk ki a rendszer lelkére, a Service-ekre. Ezek olyan állandó, injektálható szolgáltatások, amelyek az egész alkalmazást behálózzák. Vegyük sorjában mit is értünk ez alatt. Lényegében egyetlen példány kerül létrehozásra az applikáció teljes működése során, melyet a különböző osztályok használhatnak a Dependency Injection (DI) segítségével. Ez lehetővé teszi az adatok megosztását a komponensek között, illetve elérhetővé teszi a service metódusait.

Ilyen servicekbe csoportosítjuk ki például az adatbázist kezelő metódusokat vagy az autentikációért felelős műveleteket. Így bármelyik komponensben elérjük és használhatjuk ezeket.

Felépítésben szinte teljesen ugyan azok, mint egy komponens osztálya, annyi különbséggel, hogy annotálva vannak az *@Injectable*-el. Ugyan úgy rendelkeznek adattagokkal és metódusokkal, viszont őket csak egyetlen egyszer példányosítjuk.

```
@Injectable({
  providedIn: 'root'
})
export class FirebaseBaseService {

  constructor( private afs: AngularFirestore) { }

  async add(collectionName: string, data: any, id?: string): Promise<string>{
    const uid = id ? id : this.afs.createId();
    data.id = uid;
    await this.afs.collection(collectionName).doc(uid).set(data);
    return uid;
  }

  getById(collectionName: string, id: string): Observable<any>{
    return this.afs.collection(collectionName).doc(id).valueChanges();
  }

  update(collectionName: string, id: string, data: any){
    this.afs.collection(collectionName).doc(id).update(data);
  }

  delete(collectionName: string, id: string){
    return this.afs.collection(collectionName).doc(id).delete()
  }
}
```

3.3.1. ábra: Adatbáziskezelő belső service [Saját forrás]

A 3.3.1.-es ábrán látható service egy a Cloud Firestore adatbázissal kommunikáló belső szolgáltatás. Itt megvalósulnak többek között a CRUD (Létrehozás, Olvasás, Módosítás, Törlés) műveletek, mint egy vele egyenértékű DAO rétegben.

A fenti ábra nagyon jól szemlélteti, mit is értünk egy service injektálása alatt, ugyanis a servicekbe is injektálhatunk további serviceket. Jelen esetben, ahhoz, hogy használni tudjuk a Firestore adatbázist, szükségünk van a saját service-ének injektálásra. A Dependency Injection az Angular esetében a konstruktorban valósul meg és ez minden egyes komponens osztályuknál így lesz.

A servicek elkészítése nem okozott különösebb nehézséget a szakdolgozat során, mivel ezeket egységes és absztrakt szerkezeti céllal kellett megírni. Vegyük példának ez az adatbáziskezelő service-t. Itt a különböző műveleteket általánosan, vagyis bármely kollekcióra alkalmazhatónak írtam meg, így nem szükséges minden adatmodell esetén új service-t készítenünk, hanem elég csak ezzel az eggyel dolgoznunk.

3.4. Autentikációs modul

Ez az a modul, amellyel a felhasználó először találkozik amint az oldalra látogat. Itt történik a felhasználó regisztrálása és azonosítása, továbbá ő felel azért, hogy az oldalt csakis azok ériék el, akik regisztrált fiókkal rendelkeznek. Egyfajta várkapuként tekinthetünk rá, amely két elemből áll: Regisztrációs és Bejelentkezés oldalból.

3.4.1. Regisztrációs oldal

A regisztrációs oldal egy alapvető beviteli felület, ahol létrehozhatunk egy felhasználót, amely a CRM rendszerben betöltött szerepkör alapján Értékesítőnek minősül.

Különböző, főként a kommunikációt elősegítő, információk adhatóak meg. A személyes információkon túl, mint például a felhasználó kereszt- és családneve, többek között telefonszámot, illetve a vállalatnál betöltött pozíciót is társítunk a fiókunkhoz. Fontos kiemelni az e-mail és a jelszó mezőket, ugyanis ez a két fő autentikációs adat, amelyet a rendszer bejelentkezéskor ellenőriz. Az email címünk itt egy azonosító szerepet tölt be, viszont a későbbi email- vagy jelszóváltoztatáskor erre a címre küld értesítő emailt a rendszer.

A jelszót két helyen kell megadnunk, a már szinte klasszikussá vált jelszó-ellenőrzés céljából. Egy kisebb segítség a folyamatban, hogy a PrimeNg könyvtár segítségével implementált jelszóerősség-figyelő tájékoztat minket, hogy milyen erősségű a jelszónk. Az indikátor a szöveg hosszával arányosan változik.

A regisztrációs folyamat egyszerre hoz létre felhasználót a Firebase autentikációs adatbázisában és a Cloud Firestore-ban. Ezek az általam definiált két service-ben mennek végbe.

3.4.2. Bejelentkezés oldal

A bejelentkeztető oldal a rendszer alapértelmezett felülete, ami, ha nem vagyunk bejelentkezve rögtön ide navigál minket, köszönhető az Angular Routing modulnak, ami felel a rendszer teljes navigációjáért és a komponenseinek betöltéseiért.

Ez még a regisztrációnál is egyszerűbb felület, itt ugyanis csak e-mail címre és egy jelszóra van szükségünk, mivel ezek szükségesek a felhasználó hitelesítésekor. Amennyiben nem helyesen adjuk meg az adatokat hibát fog írni a felület, szigorúan az adatvédelmi irányelvek alapján, nemdeterminisztikusan jelezve azt, tehát nem tudhatjuk, hogy jelszó vagy e-mail cím helytelenségéről van-e szó.

3.5. Profil modul

Először is tisztáznunk kell, hogy két féle profilnézet van jelenleg az alkalmazásban: egy saját és egy idegen profilnézet. Értelemszerűen a saját profilnézetünkben a saját fiókunk adatait látjuk, míg az idegen nézetben egy kollégánk profilját.

A kardinális különbség a kettő között, hogy az előbbinél látjuk és tudjuk szerkeszteni az összes fiókinformációinkat, míg az utóbbinál csak a fontosabb adatokat látjuk az adott személyről, viszont jogosultságunk nincs ezek módosításához.

Szinte minden információnk elérhető és módosítható. Többek között a jelszónk is, amelynek egy külön dialógus ablakot szántam. A működése ugyan az, mint a regisztráció esetében, ugyanis itt szintén szükséges a jelszó kétszeri megadása. Ennek a folyamatnak, illetve az email változtatásnak a hatására az *authService* lép működésbe.

Az itt futó műveletek többek között a *user* és *auth* service-k segítségével hajtanak végre módosításokat. Továbbá ez az egyik olyan panel, ahol a *storageService* is részt vesz a folyamatokban, ugyanis ez felel a profilkép cseréjéért. Ahogy már említettem, a firebase szolgáltató számunkra egy tárhelyt is, ahova fájlokat tölthetünk fel. Ezt főként az applikáció profilképtárolásra használja, amelyeknek az elérési útját a profilképhez tartozó felhasználó ID-ja alapján helyezi el, vagy épp olvassa be.

Technikailag az applikációban fellelhető majdnem mindegyik beviteli mezőcsoport Angular Form-okkal valósul meg a háttérben.

```
this.form = new FormGroup ({
  email: new FormControl(this.user.email, [Validators.email, Validators.required]),
  firstName: new FormControl(this.user.firstName, [Validators.minLength(1), Validators.required]),
  lastName: new FormControl(this.user.lastName, [Validators.minLength(1), Validators.required]),
  phone: new FormControl (this.user.phone, [Validators.required, Validators.pattern("^[0-9]*$")]),
  position: new FormControl(this.user.position, [Validators.required])
})
```

3.5.1. ábra: Profil oldal form-ja módosításhoz [Saját forrás]

Úgy tekinthetünk rá, mint egy skálázható űrlap kódszerű leírása. Rendelkezik mezőkkel, amelyeknek van alapértéke és több validációs kritériuma (FormControl zárójel bal- és jobboldala). Mezőit a felületen módosítva, valós időben változtatja a form értékét is, ezeket nevezzük reaktív form-oknak (RxJs Reactive Forms). A formokat pedig adatmodellekként továbbítjuk az adatbázis felé.

3.6. Kezdőlap

A bejelentkezés után ide irányít minket a rendszer. Ez egy sokkal összetettebb, mint az eddigi oldalak, ugyanis azon kívül, hogy újrahazsnosít egy másik modult, komplexebb elemekből áll.

Két, a felhasználó számára fontos adatokat tartalmazó, panelt jelenít meg. Az egyik a felhasználóhoz rendelt aktív értékesítések előzményei, a másik pedig a felhasználó aktív, még le nem járt teendőinek listája csoportosítva. Ezekről bővebben az „3.8.1. Értékesítés oldala” alfejezetben olvasható részletes leírás, mivel értékesítési modulhoz kapcsolódnak. Előljáróban annyit érdemes tudni róluk, hogy az értékesítéseken folytatott műveletek előzményeket generálnak, amit a rendszer eltárol, ezáltal visszakövethető a folyamat alakulása a későbbiekben.

Teendők egy adott értékesítéshez vannak rendelve, amelyek szintén abban a modulban képződnek. Megjelenítésüket azért tartom fontosnak a Kezdőlapon, hogy a felhasználó bejelentkezés után kapjon egy átfogó képet az értékesítései állapotáról.

Az előző modulokkal ellentétben itt már több adatolvasási kérést is intézünk az adatbázisunk felé, így érdemesnek tartom, hogy ezekre kitérjünk részletesebben is. Egy ilyen *firebaseService* által indított olvasási kérés egy úgynevezett *Observable* típust küld vissza. Ez

a típus egy csatorna, amelyeken keresztül adatok érkeznek a komponensünkbe. Ezeknek a tulajdonsága az, hogy fel kell iratkozni rájuk, így adatváltozás esetén valós időben kapjuk kézhez az adatokat.

```
getTask(){
  this.fbService.getById("tasks",this.editId).subscribe(result =>{
    this.editTask = result
    this.form.value.title = result.title
    this.form.value.description = result.description
    this.form.value.dueTo = result.dueTo == null ? null: new Date(result.dueTo.seconds * 1000);
    this.selectedUser = this.dialog.users.find(elem => elem.id == result.responsibleId)
  })
}
```

3.6.1. ábra: Egy Teendő adatbázis olvasási művelet ID alapján [Saját forrás]

A fenti ábra szemlélteti egy Teendő modell egyszerű olvasási műveletét. Az injektált *fbService* (FirebaseBaseService) *getById()* metódusát használjuk erre célra. Az absztrakt struktúrájának hála elég csak megadnunk neki, hogy melyik kollekcióban (adattáblában) szeretnénk végrehajtani ezt a kérést, illetve, hogy melyik ID-t keressük. Ezután a visszaadott *Observable* típusára feliratkozunk a *subscribe()* metódusával. Ebben a belső scope-ban pedig amint kapunk eredményt, már rögtön dolgozhatunk is vele.

Az előzményeket hasonló módszerrel kérjük el, annyi különbséggel, hogy szűrünk a jelenlegi felhasználóra, illetve a hozzárendelt értékesítésekre aktualitás szerint sorba rendezve. A teendők esetén is ugyan ez a helyzet, csak itt a teendőhöz rendelt felhasználó szerint szűrünk, továbbá az egyszerűbb áttekintés érdekében ügyfelek szerint csoportosítjuk őket.

Rákattintva ezekre az elemekre, a hozzájuk tartozó értékesítési felületre irányít a rendszer, ezáltal azonnal el tudjuk érni az aktuális ügyeinket.

3.7. Értékesítés-áttekintő modul

Az alapvető szerepe ennek modulnak, hogy letisztultan vizualizálja az értékesítéseinket, annak érdekében, hogy egyszerű legyen a köztük lévő keresés és navigáció.

Ezt alapvetően a PrimeNg szervezeti diagramstruktúrájával (Organization chart) valósítom meg, amelynek gyökerei a vállalatok, avagy az ügyfelek, leveleik pedig a hozzájuk tartozó értékesítések, attól függően, hogy aktív vagy lezárt a státuszuk.

Egy kifejezetten könnyen áttekinthető megjelenítést eredményez ez számunkra, ahol egyértelműen látszódik, mi mihez kapcsolódik, viszont nincs túlinformálva sem a felhasználó, csupán azok az adatok állnak rendelkezésre, amelyek feltétlen szükségesek egy értékesítés beazonosításakor.

Lehetőség van váltani két nézet között. Egyik nézetben a *Felmérés és Folyamatban van* státuszú értékesítések vannak, míg a másikban a *Lezárt* státuszúak. Ezt azért tartottam nagyon fontosnak, hogy legyen egy jelenlegi és egy implicit archívum nézet. Így a felhasználó rögtön szembesül a jelenleg is futó értékesítésekkel, viszont, ha szeretne kutatni a régebbiek között akkor a másik nézetben meglehet. Ennek hála nem lesz túlcsondultva értékesítésfajtákkal a diagramunk.

Egy kisebb jelmagyarázatot is csatoltam a panel aljára a könnyebb értelmezhetőség miatt, mivel az értékesítések, a sorszámuk mellett, egy színt is kaptak háttérnek a státuszuk alapján. Sokkal letisztultabb és esztétikusabb megoldás, mintha csak kiírtuk volna őket. Ezen kívül ezek az elemek főként navigációs célt szolgálnak, ugyanis, ha megnyomjuk bármelyik elemet a diagramban, akkor a hozzá kapcsolódó felületre navigál minket.

Az egyik legnagyobb kihívás ebben modulban ennek adatstruktúrának az összegyűjtése, válogatása és rendezése volt. Tulajdonképpen ez egy fa struktúra (TreeNode) sorozat. Leegyszerűsítve, minden vállalat/ügyfél egy külön fa, amelyek egy tömbben egymás után helyezkednek el. A fő nehézség ezeknek az effektív összegyűjtése volt, illetve az említett elméleti struktúra megértése és átalakítása futtatható kódra. Végeredménye végül egy terjedelmes rendező-szelektáló metódus lett.

Ebben a modulban hozhatunk létre új értékesítéseket is egy új dialógus segítségével. Ez az eddigiekhez képest több beviteli mezőt vonultat fel egyetlen felületen, de nem kell tőle megijedni. A nagyszerűsége ennek a panelnek, hogy ha már létezik a vállalat, akinél értékesítést szeretnénk indítani, akkor nem kell beírni újra az összes céginformációt, hanem egyszerűen elég kiválasztanunk a céget a listából.

Ezt a listát beviteli mező valós idejű figyelésének hála lehetőség van gépelés közben is elérni, ugyanis az ott lévő szöveg alapján szűrést hajt végre a cégek listájában és eredményként egy szűkebb, a rész-szöveget tartalmazó cégneveket listázza ki. Kiválasztva az egyiket betölti a formunk elemeibe a cégadatokat.

```

choosedComp(event){
  let company = this.companies.find(elem => elem.company.id == event.id)

  this.companyForm.value.name = company.company.name
  this.companyForm.value.ceoName = company.company.ceoName
  this.companyForm.value.email = company.company.email
  this.companyForm.value.phone = company.company.phone
  this.companyForm.value.taxNumber = company.company.taxNumber
  this.companyForm.value.webpage = company.company.webpage
  this.companyForm.value.address = company.company.address

  this.personForm.value.firstName = company.contact.firstName
  this.personForm.value.lastName = company.contact.lastName
  this.personForm.value.position = company.contact.position
  this.personForm.value.email = company.contact.email
  this.personForm.value.phone = company.contact.phone

  this.fbService.getFilteredByIdList("sales",company.company.id,"companyId").subscribe(result =>{
    result.forEach(element => {
      if(Number(element.saleId) >= Number(this.biggestId)){
        this.biggestId = element.saleId
      }
    });
  });
}

```

3.7.2. ábra: Kiválasztott vállalat információinak betöltése [Saját forrás]

A 3.7.2.-es ábrán látható kódrészlet abban az esetben fut le amikor kiválasztunk egy már létező céget a legördülő listából. Itt feltölti a két formot az adatokkal, illetve megkeresi a céghez tartozó legnagyobb Értékesítési ID-t, amit új értékesítés létrehozása esetén 1-gyel megemel, így biztosítva, hogy a legnagyobb ID-ú értékesítés lesz a legújabb az adatbázisban.

3.8. Értékesítés modul

Ebben a modulban teljeseedik ki igazán az alkalmazás. Ez egy három-oldalas felület, ahol a cég- és ügyfélinformációk találhatóak, illetve itt érjük el magát az értékesítési nézetet, vagyis az alkalmazás lényegi elemét. A fenti fülek használatával navigálhatunk az oldalak között.

3.8.1. Értékesítés oldala

Az egyszerűség kedvéért tartalmilag három részre bontom ezt az alfejezetet, mivel az oldal maga is három részből áll. Egy teendők, egy előzmények és egy értékesítési folyamat panelből.

Az Értékesítés panel talán a legfontosabb mind közül. Ez egy komplex folyamatkövető alfelület, ahol az értékesítés minden információját láthatjuk, beleértve az aktuális státuszát, ugyanis a státuszok alapján tagolódik a panel. Attól függően, hogy épp milyen állapotban van (Felmérés, Folyamatban van vagy Lezárt) úgy válik aktívvá a hozzátartozó státuszpanel. Szürkék a nem aktív státuszú panelek és színes a jelenlegi státuszt szimbolizáló panel. Ilyenkor az adott státusz előtti összes panelt láthatjuk, míg az utána lévő(ke)t nem. Például, ha egy értékesítés státusza *Folyamatban van*, akkor ez a panel lesz a színezett, míg a felette lévő *Felmérés* státuszpanel csak lenyitható lesz viszont nem kattintható, illetve a *Lezárt* státuszpanel le sem lesz nyitható, mivel ott még nem tartott az értékesítés. Lényegében ennek a célja az, hogy a korábbi státuszok bevitt információkat láthassuk, viszont a későbbi státuszokét ne, hisz még ott nem tartottunk az értékesítésben, ezért azok nem is elérhetőek.

Ezek a státuszpanelek beviteli mezőket tartalmaznak, amelyek különböző információkat szolgáltathatnak az értékesítőnek vagy később például egy marketing csoportnak. A Felmérési státuszban azon kívül, hogy hozzá tudjuk rendelni felelősként egy adott kollégát, információkat adhatunk meg az ügyfélről, mint például, hogy honnan ismer minket, vagy honnan hallott rólunk, illetve, hogy milyen típusú ügyfél (Egyszer, Állandó...stb). Ezen kívül általános információkat is megadhatunk, illetve az ügyfél aggályait is feltüntethetjük, amelyek különösen nagy fontossággal bírnak egy értékesítés során.

A Folyamatban lévő értékesítéseknél hasonló a helyzet, viszont itt már egy bizonyos szintű predikciót is meg kell adnunk az alapján, hogy mikor és milyen bevétellel fog valószínűleg teljesülni az értékesítés vagy hogy mi a várt eredmény. Ez azt az állapotot tükrözi amikor egy értékesítés már bemutató fázisban van és rendelkezünk valamilyen jellegű, akár implicit, ügyfélmegállapodással. Feltölthetünk egy ide kapcsolódó dokumentumot is, amennyiben van olyan, amelyet szeretnénk elmenteni ehhez az értékesítéshez. Például ilyen dokumentum lehet egy szerződés vagy prezentáció is. Ahogy azt már korábban írtam, a Firebase tárhely szolgáltatását eddig a pontig csak profilképek tárolására használtuk, itt viszont bevezetjük a fájlfeltöltés opciót is, amely szintén ezt az erőforrást használja, csak másik elérési útvonalon tárolja a fájlokat, viszont ugyanúgy ID-val (SaleID) azonosítja a dokumentumokat.

A Lezárt státusz két féle lehet: Sikeres (Zöld) vagy Sikertelen (Piros), értelemszerűen az értékesítés sikerességétől függően. Ezen a felületen a lezárás dátumát és okát adhatjuk meg, sikeresség esetén ez kiegészül a valódi bevétellel is.

A folyamatkövetés működési elve a státuszok köré épül, ezért is bírnak nagy jelentőséggel a rendszerben. Bizonyos elemek megjelenítése, illetve elérhetősége nagy százalékban aszerint változik, hogy az adott értékesítés milyen státuszban van.

Ezen a paneleken túl az oldal tetején található egy státusz folyamatábra is, amely ugyanezt a színezési logikát alkalmazza, viszont ez pusztán státuszkövetésre szolgál, ennek ellenére kifejezetten esztétikus.

Két féle művelet valósul meg gombok segítségével ezeken a státuszpaneelen, az egyik a státuszváltás a másik pedig a mentés. Mind a két művelet generál egy előzményeseményt, amennyiben történt érdemi változtatás. Státuszváltáskor ez a feltétel mindig teljesül.

Ha már az értékesítési előzményeknél tartunk, vegyük át milyen elemei vannak egy ilyen adatnak. Természetesen tartalmaznia kell, hogy: mi változott, ki változtatta és hogy mikor változtatta. Ezeket a baloldali legalsó panelen jelenítjük meg. Itt megtalálható az összes műveleti előzmény, ami ehhez az értékesítéshez kapcsolódik, időrendi sorrendben.

A panelen fel van tüntetve a személy profilképe és neve, legfelül az előzmény típusa, alatta a változtatás tartalmi része, illetve a sarokban az előzmény dátuma. Természetesen a profilképre kattintva átirányít minket a rendszer az adott személy profil oldalára. Fontosnak tartottam, hogy szerepeljen egy ilyen jellegű vizualizált adatfolyam, mert vannak olyan forgatókönyvek, ahol valóban jól jönnek múltbéli információk áttekintése, amelyek időközben megváltoztak.

A legutolsó elem, amiről beszélnünk kell ezen az oldalon, azok a Teendők komponens. Itt a jelenlegi értékesítéshez kapcsolódó teendők vannak felsorolva aktualitási sorrendben. Hasonló a megvalósítása, mint az előzményeké, azzal a különbséggel, hogy ezek törölhetőek és szerkeszthetőek is, mivel a létrehozásuk manuálisan történik. Itt a legfelül -az előzményekkel ellentétben- a teendő címét láthatjuk mellette pedig a határidejét. A panel jobb felső sarkában lévő gombra kattintva új teendőt hozhatunk létre. A teendőket is hozzárendelhetjük egy kollégánkhoz, akár csak az értékesítési panelnél, illetve itt is elég csak egy legördülő listából kiválasztanunk őket. Szerkesztéshez a listázott teendők egyikénél lévő kék ceruza ikonra kattintva tudjuk szerkeszteni azt, természetesen csak a hozzánk kapcsolódókat.

Technikai részről ez egy beágyazott komponens, amely azt jelenti, hogy szülő-gyerek kapcsolatban áll az őt beágyazó komponenssel. Mint ahogy a Kezdőlapon, itt is a gyerek

szerepet tölti be. A két komponens úgynevezett dekorátorokkal kommunikál egymással, vagyis így oszthatnak meg egymással adatot. Ezek az *@Input* és az *@Output* dekorátorok.

```
export class TaskComponent implements OnInit {  
  @Input() dialog = {show: false, saleForm: {} as ISale , users: []}  
  
  @Output() closeEvent = new EventEmitter<boolean>();  
}
```

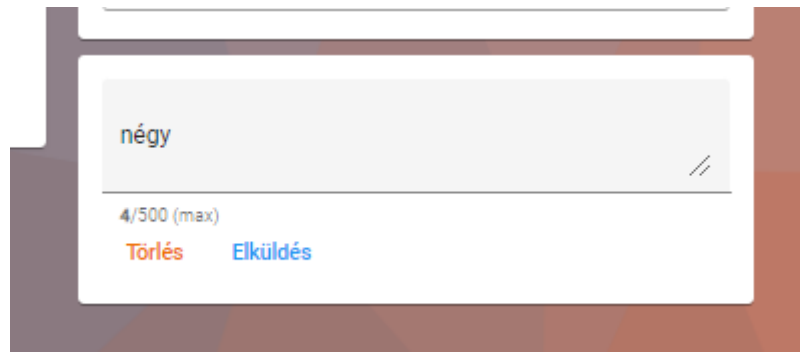
3.8.1.1. ábra: Dekorátorok [Saját forrás]

A gyerek komponensben deklaráljuk őket, tehát ebből a szemszögből, az *Input* adatot ad neki, az *Output* pedig adatot küld vissza a szülőnek. A szülő komponens *.html* fájljában ágyazza be ezt a komponenst, amelyet ellát a kívánt információkkal, amit a gyerek az *@Input* segítségével fogadhat valós időben.

3.8.2. A cég- és kapcsolattartó oldala

A felső füleken tovább navigálva eljuthatunk a Cégoldalra és a Kapcsolat oldalra. A két oldal felépítésében szinte ugyanaz, csak tartalmilag különböznek. Az alapstruktúra az, hogy az értékesítések mindig egy adott céghez kapcsolódnak, illetve mindegyik céghez továbbá kapcsolódik egy kapcsolattartó személy. Ezért beszélhetünk külön Cégoldalról és Kapcsolattartói oldalról.

A Cégoldalon találjuk a vállalat alapvető információit, mint például székhely vagy adószám. Lényege, hogy megjelenítse a szükséges információkat, amelyeket szerkeszthetünk is, ha időközben szükségünk van rá. Ezek mellett lehetőségünk van megjegyzéseket hagyni a cégoldalokon, hasznos információkkal segítve a többi értékesítőt. Ez a jobb oldali panelon kapott helyet, ahol listázva látjuk a kommenteket időrendben. Egy új megjegyzés írását az alján lévő beviteli szekcióban tehetjük meg. Ezeket kommentek aztán lehetőségünk van módosítani és törölni is, de jogosultságunk csak a saját megjegyzéseinkre terjed ki.



3.8.2.1. ábra: Megjegyzés írása [Saját forrás]

Ezeknél a beviteli mezőknél, mivel meghatároztunk egy felső korlátot (max. 500 karakter), egy karakterszámlálót is implementáltam, hogy a felhasználónak rálátása legyen erre (3.8.2.1. ábra).

A Kapcsolat oldal, ahol a kapcsolattartó személy adatait találjuk, ikertestvére az előzőnek. Ugyan az a funkcionalitása és logikája, mint a Cégoldalnak. Erre azért volt szükség, hogy a személyhez kapcsolódó információk elkülönítve legyenek jelen a két felületen, gondolok itt főként a megjegyzés információkra. Ahogy a dolgozatom elején is említettem, fontos elkülöníteni a céges, száraz adatokat, a személyes, egyéni információktól.

3.9. Termékek oldal

Ez a felület nem feltétlen része egy CRM célszoftvernek, viszont a rendszer kontextusba helyezése céljából nélkülözhetetlennek éreztem egy ilyen oldal létrehozását is. Az értékesítési logika fejlesztése közben tudatosult bennem, hogy maga az értékesítés tárgya sehol sincs definiálva, vagyis például egy értékesítés közben honnan tudjuk a rendszerből, hogy mit is értékesítünk?

Szigorúan csak a rendszer kiterjedésében szemlélve a problémát, sehol nincs lehetősége a felhasználónak megtekinteni a termékeket. Természetesen ez egy CRM rendszernél nem feltétlen baj, hisz nem ez az elsődleges rendeltetése, illetve a megrendelő határozhatja meg az összetételét.

Ennek ellenére én úgy döntöttem, hogy létrehozok egy ilyen felület, amely listázza a már meglévő termékeket, illetve ahol hozzáadhatunk újakat, törölhetünk és módosíthatunk elemeket. Egy ERP integrált rendszereként a CRM ilyen jellegű funkcióját egy másik rendszer megoldaná, így elég lenne csak onnan elkérni az adatokat, de jelen esetben egy célszoftverről beszélünk, ahol erre nincs lehetőség.

A fő elem itt egy Angular Materials táblázat, amelynek elemei az oszlopok alapján sorba rendezhetőek. Jelenleg a táblázat sora az alapinformációkat jeleníti meg egy termékről, amely most egy mezőgazdasági-építőipari hálókat forgalmazó vállalat termékstruktúráját mintázza.

Ezen kívül egy dialógus ablakot is létrehoztam, hogy az adott táblázat elemét részletesen is lehessen ábrázolni. Továbbá itt helyeztem el a törlés és módosítás funkciókat is. A törlést annyival egészítettem ki, hogy szükségessé tettem egy ellenőrzést is, aminek lényege, hogy a törlendő elemet csak akkor törölhetjük, hogy ha a megadott helyre beírjuk a tárgy kódját, ami egyezőség esetén engedélyezi a törlést, így meggátolva azt, hogy véletlenül eltüntessünk egy fontos tételt a terméklistából.

```
getRow(row: IProduct){
  const dialogRef = this.dialog.open(ProductDialogComponent, {
    width: '500px',
    data: {product: row}
  });

  dialogRef.afterClosed().subscribe(async result => {
    if(result == "delete"){
      this.fbService.delete("products",row.id);
      this.messageService.add({severity:'success', summary:'Sikeres törlés'});
    }
    if(result instanceof Object){
      this.fbService.update("products",result.id,result)
      this.messageService.add({severity:'success', summary:'Sikeres mentés'});
    }
  });
}
```

3.9.1. ábra: Termék dialógus megnyitása [Saját forrás]

A 3.9.1.-es ábrán látható kódrészlet egy adott sorra való kattintáskor lép működésbe. Ez jeleníti meg, avagy hívja meg a dialógus ablakot, illetve várja annak bezárását. A dialógus bezárásakor, attól függően, hogy törlünk, mentünk vagy egyszerűen bezárjuk, különböző feladatokat futtat le. Az említett két művelet esetén egy értesítő üzenetet is megjelenít (*messageService*), amelyről a későbbiekben lesz szó (3.11.2. fejezet).

Új elem hozzáadására is van lehetőség, ami a táblázat egy sorát mintázza. Ennek a kialakítása igazán trükkös volt, főként az, hogy ez a panel teljesen hozzásimuljon a táblázat aljához, bármekkora is az. A bevétel úgyszintén Form-al valósul meg, illetve validációs feltételeket is alkalmaz.

3.10. Statisztikai oldal

A rendszer elemzési funkciói itt vizualizálódnak a PrimeNg különböző diagramjai segítségével. Két elemzési aspektusra bontottam ezt az oldalt: Pénzügyi és Értékesítési.

Az Pénzügyi statisztika lényege, hogy egy átfogó képet kapjunk arról, milyen pénzügyi predikciós és megvalósult eredmények történtek az adott hónapban. Összegyűjti az értékesítések várható bevételeit a várható lezárási dátumok alapján, illetve ugyan ezt elvégzi a már eredménnyel rendelkező értékesítésekkel is. Az így kapott adatokból kimutatható, hogy az adott hónap várható bevételéből mennyi valósult meg, vagy hogy melyik volt az a hónap, ahol a legnagyobb eltérés volt. Ezeket a felületen diagramban vizualizáljuk.

Az Értékesítési statisztika egy arányaiban tartalmasabb és látványosabb felület. Ugyanis az értékesítés adatmodell lényegesen nagyobb, mint bármelyik a rendszerben ezáltal több adat áll a rendelkezésünkre egyszerre, amelyekkel jó pár analitikai kombinációt megvalósíthatunk.

Jelenleg három ilyen elemzés van ezen az oldalon. Az első az adott hónapban létrejött értékesítések státusza alapján épít egy fánk-diagramot, attól függően, hogy a négy típusú státusból mennyi van jelenleg. Mellette helyezkedik el egy termékcsoporthoz statisztika, amiről leolvasható, hogy a kiválasztott hónapban létrejött értékesítések, milyen termékcsoportokat érintettek. Ez a radar-diagram tökéletes lehet a szezonális vizsgálatára vagy egyéb termékjelentések támogatására. A legalján pedig az értékesítőkre lebontott értékesítési sikeresség jelenik meg oszlopdiagramként, a kiválasztható hónap mellett. Itt látható, hogy az adott személy milyen arányban végzett sikeres vagy sikertelen értékesítést. Ez egyéni teljesítménymérést tesz lehetővé a cég számára.

Természetesen ezek mind tovább bővíthetők, illetve kombinálhatóak a jövőben. Nagyon jól szemlélteti, hogy milyen elképesztő Angular könyvtárak léteznek az adatok vizualizálásra.

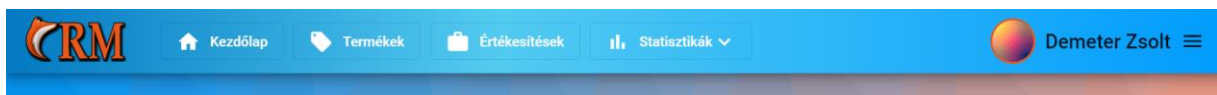
3.11. Egyéb elemek

Ebbe a fejezben olyan elemek kerültek, amelyek főként megosztott elemek, tehát szinte mindenhol megjelennek a rendszerben.

3.11.1. A menüsáv

A menüsáv, mint minden weboldalon, elsősorban navigációs szerepkört tölt be, illetve egyéb információk megjelenítésére szolgál. Jelen esetben a bejelentkezett felhasználó nevét és profilképét láthatjuk rajta jobb oldalt, illetve a navigációs gombokat bal oldalt.

Gombokból négy található, melyek közül a Statisztikák gomb egy legördülő listát jelenít meg, a már tárgyalt, két statisztikai nézetel. Ezen kívül a felhasználónév mellett jobbra is található egy menügomb, ahol a saját profilunkat érhetjük el és ahol kijelentkezhetünk.



3.11.1.1. ábra: Az applikáció menüsávja [Saját forrás]

Az alkalmazásban használt ikonok forrása: [\[https://www.iconfinder.com\]](https://www.iconfinder.com)

3.11.2. UI értesítések

Ezek olyan felugró kis értesítésablakok, amelyek bizonyos műveletek (regisztráció, módosítás, létrehozás... stb) során keletkeznek és a jobb felső sarokban jelennek meg. Főként egy folyamat sikerességét jelzik, hogy értesüljünk a műveletek befejezéséről. Ez úgyszintén egy PrimeNg elem, amelyet a gyökérkomponensbe helyezünk és a MessageService segítségével érhetünk el.

```
} as IUser).then(res => {
  this.messageService.add({severity:'success', summary:'Sikeres regisztráció'});
  this.router.navigateByUrl("/login")
})
```

3.11.2.1. ábra: Regisztrációs értesítés [Saját forrás]

A fenti ábrán látható regisztráció után lefutó kódrészlet, jól szemlélteti ennek működését. Miután az service injektálás megtörtént, egy egyszerű `add` metódussal hozzáadhatunk a gyökérkomponensben lévő `<p-messages>` elemhez egy új értesítést, ami jelen esetben egy regisztrációs értesítés lesz (3.11.2.1. ábra).

4. A FOX-CRM FUTTATÁSA

4.1. Lokális futtatás

Mindenekelőtt szükséges a 3.1.4.-es fejezetben felsorolt -megfelelő verziójú- elemek telepítése. Ezután a forráskódokat tartalmazó mappában megnyitott terminál parancssorában az **“npm install”** utasítás kiadása után letöltődnek és települnek a szükséges függőségek. Számítógépenként eltérhet ennek ideje, de tapasztalatok alapján pár perc alatt végez.

A telepítés után az **“ng serve”** parancs kiadásával futtathatjuk az applikációt. Egy lokális webszerveren fogja elindítani az alkalmazást, amelyet alapértelmezetten a *localhost:4200* címen érünk el egy böngészőben.

4.2. Firebase Hosting

Lokális futtatáson túl, az interneten keresztül is elérhetjük a webalkalmazás egy példányát. Ahogy azt a Firebase tulajdonságainál tárgyaltuk, egy Hosting szolgáltatást is biztosít számunkra a Firebase. Lényegében egy dedikált webszervert nyújt a számunkra, saját domain névvel, ahova telepíthetjük a webalkalmazásunkat.

Ezáltal nem csak nekünk, de egyszerre több embernek is elérhető lesz a rendszer. Továbbá, a Firebase felületén követhető és elemezhető a webszerver működése, ami a beépített monitorozásnak köszönhető.

Jelenleg a telepített online webalkalmazás a <https://fox-crm-511f9.firebaseio.com/login> és a <https://fox-crm-511f9.web.app/login> címeken érhető el.

Egy szemléltető példafelhasználó, amivel bejelentkezhetünk:

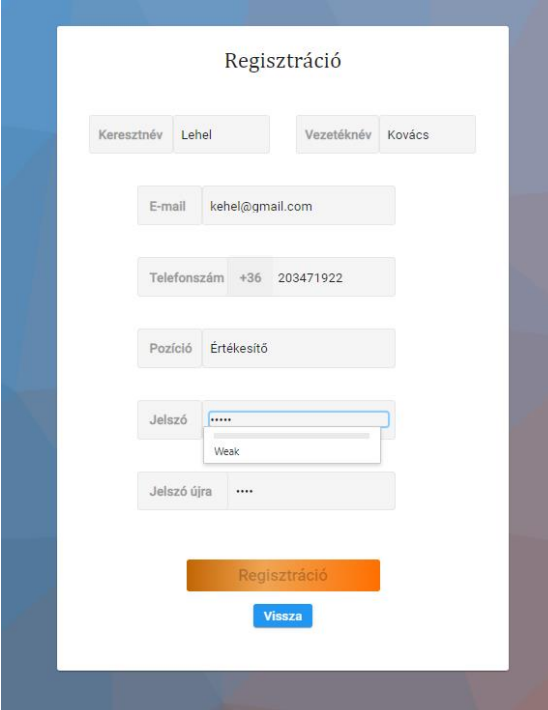
- E-mail: *zsolf57@gmail.com*
- Jelszó: *admin2*

5. A FOX-CRM BEMUTATÁSA

5.1. Autentikáció

A regisztráció és a bejelentkezés egy egyszerű dialógus ablakban jelennek meg és főként terjedelmükben térnek el.

A regisztráción a fiók létrehozásához szükséges minden információ beviteli mezője megtalálható. Ha minden adatunk sikeresen megfelel a kritériumoknak akkor elérhetővé válik a regisztrációs gomb, mellyel létrehozhatjuk a fiókunkat. Ha ezzel a lehetőséggel nem kívánunk élni akkor a Vissza gombbal újra a Bejelentkezés oldalra navigálhatunk

The image shows a web-based registration form titled "Regisztráció". It contains several input fields: "Keresztnév" (First Name) with the value "Lehel", "Vezetéknév" (Last Name) with the value "Kovács", "E-mail" with the value "kehel@gmail.com", "Telefonszám" (Phone Number) with the value "+36 203471922", "Pozíció" (Position) with the value "Értékesítő", "Jelszó" (Password) with a masked input "*****" and a dropdown menu showing "Weak", and "Jelszó újra" (Repeat Password) with a masked input "****". At the bottom, there is an orange "Regisztráció" button and a blue "Vissza" button.

5.1.1. ábra: Regisztrációs felület [Saját forrás]

A bejelentkezés ennél jóval szerényebb beviteli lehetőséggel rendelkezik, hisz csak a hitelesítés a célja. Ezért összesen két információ megadására van szükség: a megfelelő e-mailre és a jelszóra. A Bejelentkezés gombra kattintva pedig elérhetjük az alkalmazást.

5.2. Profilok kezelése

Ahogy már a „3.5. *Profil modul*” fejezetben említettem megkülönböztetünk egy saját és egy idegen profil nézetet.

5.2.1. ábra: Saját profilnézet [Saját forrás]

A felület felosztható egy jobb- és egy baloldali részre. A baloldal szemlélteti, hogy mit fog látni egy másik felhasználó a profilunkból, illetve itt cserélhetünk profilképet, míg a jobboldal az adatmódosításokért felel. Az idegen nézet csak a baloldali panelből áll, míg a jobboldali panel csak a saját profilunkon elérhető. Itt az Új jelszó gomb egy felugró dialógus ablakot jelenít meg, amellyel jelszóváltoztatás hajtható végre. A Mentés gomb a nevéből eredően elmenti a változtatott adatokat.

5.3. Értékesítések listája

A már tárgyalt cégek és értékesítések fastruktúrájú diagramja található itt. Rájuk kattintva pedig az adott elemre navigálhatunk. Az egyes fák gyökerei a vállalatok, míg a leveleik az értékesítések.

A panel bal felső sarkában lévő gombbal válthatunk az aktív és a lezárt nézet között.



5.3.1. ábra: Értékesítések diagramja [Saját forrás]

A jobb felső sarokban található az *Új Értékesítés/Vállalat* gomb, amellyel értelemszerűen új Értékesítést hozhatunk létre egy már meglévő vagy új vállalat számára.

Vállalat neve melletti mezőbe kezdjük el begépelni a kívánt cégnevet, aminek eredményeképp egy legördülő lista jelenik meg az egyező cégnevekkel, amelyekre, ha rákattintunk akkor betölti a cég összes információját az itteni mezőkben. Ha ennél gyorsabb megoldást szeretnénk akkor a legördülő sáv kék ikonjára kattintva elérjük a teljes céglistát.

5.3.2. ábra: Új értékesítés/Vállalat hozzáadása [Saját forrás]

Amennyiben új céget hozunk létre ott természetesen szükséges ezek manuális kitöltése. A száraz adatok mellett kiválaszthatjuk, hogy az új értékelésünk melyik kollégánkhoz tartozzon, illetve milyen kezdeti státusszal rendelkezzen. Az indítás gombra kattintva pedig el is készül az új, még üres értékesítésünk.

5.4. Értékesítéskövetés

Az értékesítés folyamatának követése az értékesítés oldal jobb oldalán található. Ez egy lefelé bővülő státuszpanelek csoportja, ahol az adott értékesítés jelenlegi státusza van kiemelve.

Alapjáraton a Felméréssel kezdődik egy ilyen folyamat. Itt megadhatjuk a kért adatokat változatos beviteli módszerekkel. A Mentés gombbal elmentjük, a Státuszváltással pedig a következő státuszba helyezzük az értékesítést. Amennyiben már itt zsákutcába fut ki a folyamat, abban az esetben azonnal le is zárhatjuk az Értékesítés lezárása gombbal.

A következő a Folyamatban lévő státusz, ahol a korábbi fejezetben említett predikciókat kell megadnunk az adott értékesítés lezárását és bevételét illetően. Kiválaszthatjuk még egy listában a termékcsoportokat, illetve módosíthatjuk az előző státuszban kitöltött aggályok mezőt abban az esetben, ha ez változott. Itt is hasonló a gombkiosztás, annyi különbséggel, hogy megjelenik a Vissza gomb, ami nevéből adódóan az előző státuszra ugrik.

A Lezárt státusz az utolsó a sorban. Szerényebb beviteli mező mennyiséggel rendelkezik, mivel itt csak a valódi lezárás dátumát és a -sikeresség függvényében- valódi bevételt kell megadnunk.

5.4.1. ábra: Értékesítési felület [Saját forrás]

Az oldalon megtalálhatóak továbbá az Állapotsáv legfelül, a Teendők közvetlenül alatta és az előzmények a legalján. Az állapotsávval és az előzményekkel közvetlen interakciónk nincs, ellenben a Teendők szabadon létrehozhatóak, módosíthatóak és törölhetőek.

A fenti menügombokkal válthatunk a vállalati és kapcsolattartói oldal között, melyek információkat tartalmaznak az adott entitásról.

5.5. Termékek

Itt listázva látjuk a termékeket, táblázat formájában. Az elemek átrendezhetőek, ha az oszlopok nevére kattintunk, ahol növekvő, illetve csökkenő rendezést is választhatunk.

Ezen kívül, kiválaszthatunk egy tetszőleges terméket, ami egy új dialógus ablakot nyit meg, ahol összesítve látjuk a termékinformációkat, illetve látható válik a termék leírása. Ezt ha szeretnénk módosíthatjuk, illetve törölhetjük. Ahhoz, hogy törölni tudjunk a megjelenő ellenőrző mezőbe bele kell írunk a tárgy kódját, és csak akkor válik lehetségessé a törlés.

Termékkód	Név ↑	Szín	Méret	Anyag	Ár
E-AL-CS-E01	Alumínium cső	ezüst	1x2m	Alumínium	10000 HUF
M-AR-H-Z01	Árnyékoló háló	fekete	1,2x50m	polipropilén	15000 HUF
X-CAS1-22	Fényő háló	kék	1x2	polietilén	123 HUF
CS-FO-H-F01	Fokhagyma háló	fehér	0,3x1000m	polipropilén	10000 HUF
F-HU-R-F01	Hüllőterelő rács	fekete	1x24m	polietilén	12000 HUF
E-MA-H-N01	MASNET 14	narancs	1,2x50m	polipropilén	30000 HUF
M-UB-H-Z01	Uborkaháló	zöld	1,7x1000m	polipropilén	60000 HUF
(Termékkód)	(Név)	(Szín)	(Méret)	(Anyag)	(Ár) HUF

Mégsem Hozzáadás

5.5.1. ábra: Terméktáblázat és új elem hozzáadása [Saját forrás]

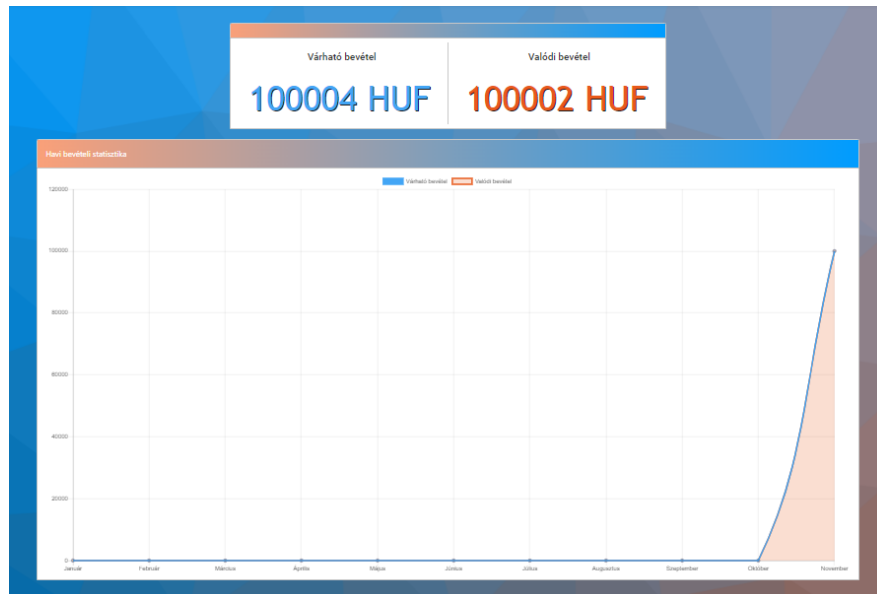
A táblázat jobb alsó sarka alatt látható kék “plusz” (+) ikon megnyomásakor egy új termék bevitelére szolgáló mezőcsoport jelenik meg a táblázat alján. A leíráson kívül minden adat megadható.

5.6. Statisztikák

A FOX-CRM jelenleg értékesítési és pénzügyi statisztikai kimutatásokkal rendelkezik. A pénzügyi statisztikáknál a bevétel a fő elemzési szempont, míg az Értékesítésnél a hozzá kapcsolódó egyéb adatok.

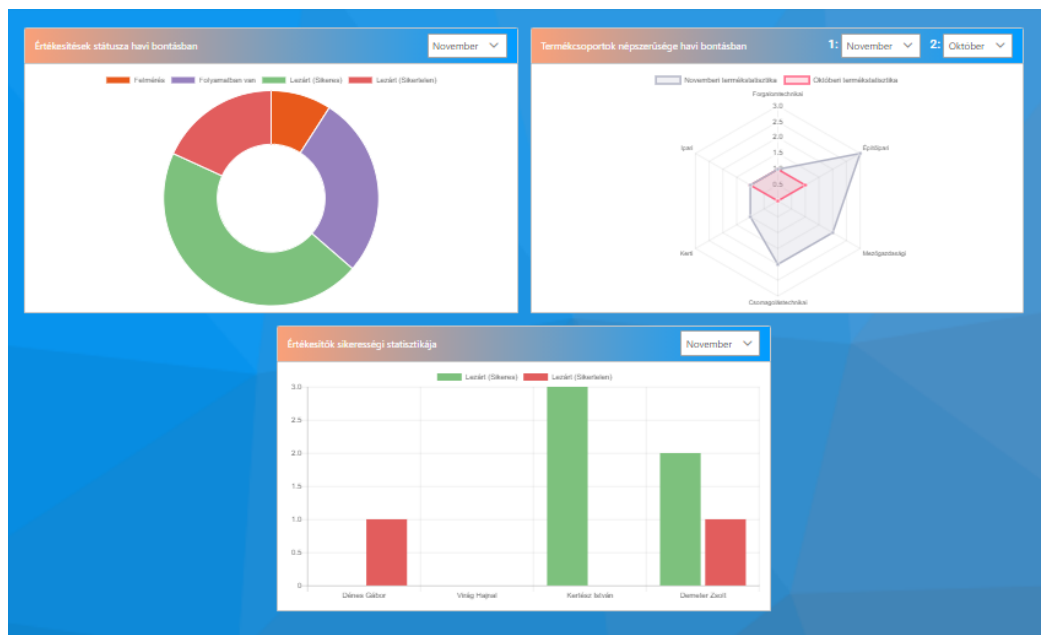
A két nézet között a menüsávban lévő Statisztikák gomb hatására lenyíló listával tudunk váltani.

A pénzügyi statisztika oldal tetején a jelenlegi hónap várható és valódi bevétele látható, alatta pedig egy vonaldiagramon hónapokra lebontva jelenik meg ez két érték alakulása.



5.6.1. ábra: Pénzügyi statisztika [Saját forrás]

Az értékesítési statisztika három diagramból áll: Egy státusz-elemzés, egy termékcsoport-elemzés és egy értékesítő-összelemzés. Ezeknek a havi eloszlását változtatni is tudjuk a panelek tetején lévő legördülő listával.



5.6.2. ábra: Értékesítési statisztika [Saját forrás]

6. Bővítési lehetőségek

Az alkalmazás több szempontból is bővíthető. Egyrészt az igények alapján változtathatóak az egyes elemei, másrészt új elemek hozzáadására is van lehetőség.

Sok ötlet felmerült a szakdolgozat során, amelyek megvalósítására nem jutott idő, főként, mert nem nélkülözhetetlen funkciók. Ilyen például egy Naptár widget.

A naptár widget kifejezetten hasznos lenne ehhez az ügyviteli rendszerhez, hisz a határidők és teendők áttekintése egy naptáron sokkal effektívebb időbeosztást eredményez számunkra. Ezt még annyival egészíteném ki, hogy összeköthető lenne a Google Naptárral, mivel egyre sűrűbben használják, illetve egyre több program menti automatikusan ebbe a naptárba az eseményeket, így célszerű számunkra is egy ilyen funkciókiegészítés

Egy másik hasonló, értékesítő támogató, funkció a termékajánlási algoritmus megvalósítása. Egy több terméket felvonultató cég számára ez nagyon hasznos funkció lehet. Az alapvető koncepció az, hogy az ügyfél által korábban megvásárolt termékek tulajdonságai alapján - súlyozva- más hasonló termékeket ajánl az értékesítőnek a felmérési fázisban. Ilyenkor dönthet úgy a munkatárs, hogy él az ajánlás adta információval és felterjeszti az ügyfélnek, ezáltal potenciálisan nagyobb értékesítési forgalmat generálva.

Tervben lévő kiegészítő funkciók közé sorolható egy Google account összeköttetés. Ez szorosan kapcsolódik a Google naptárhoz is természetesen, viszont az indíttatás onnan ered, hogy a Firebaseben erre az autentikációs típusra van is lehetőség. Jellemző, hogy a vállalatok a Google My Business segítségével kezelik az alkalmazottakat, vagyis például minden alkalmazottnak saját céges Google fiókja van.

A felsorolt funkciókon túl további ERP alrendszerek tulajdonságait is bele lehet integrálni az alkalmazásba, amelyre nagyon jó példa a már létező termékek modul. Ezt továbbfejlesztve egy egyszerűbb termékmenedzsment rendszert is megvalósíthatunk a szoftveren belül.

7. Összefoglaló

Mindent egybevetve, az általam egyedül fejlesztett FOX-CRM teljesíti mindazon elvárásokat, amelyeket egy ügyfélkapcsolat-kezelő rendszer felé támaszt egy felhasználó és rendelkezik minden olyan tulajdonsággal, amellyel egy modern ügyviteli alkalmazásnak szüksége van.

A felhasználókezelésének hála, több egymástól független ember tud egyidejűleg, jól elkülöníthető munkát végezni. Ezen sajátosság részét képezi a személyre szabhatóság, amelyhez sorolható a saját profilkép beállítása vagy az egyéni teendők kezelése.

Az értékesítés hatékony követésére és támogatására alkalmas webapplikáció interaktívan ábrázolja az értékesítések folyamatát. Igényesen és egyértelműen informálja a felhasználót az aktuális üzleti lehetőség státuszáról és információiról. A folyamatok követésében értékesítési előzmények vannak a felhasználó segítségére, amelyről leolvasható ki, mikor és mit módosított. A létrehozható teendőkkel gyorsan képet kaphat a felhasználó, hogy milyen sürgős, illetve prioritást élvező értékesítési feladatokkal kell foglalkoznia az adott napon.

Egyedi ábrázolási megoldásként az összes értékesítés egyszerre megtekinthető egy fastruktúrájú diagramon, ami nem csak könnyíti vállalatok értékesítéseinek áttekinthetőségét, de rendkívül esztétikus is.

Egy átlagos CRM rendszer eszköztárát nem képző, de itt mégis jelenlévő elem az integrált termékinformációk. Ennek hála átfogó képet kaphat a felhasználó, hogy az értékesíteni kívánt termékekről, amelyeket egy interaktív táblázatban kezelhet.

Analitikai megoldásokat is kínál a FOX-CRM, ugyanis elérhetőek különböző statisztikai kimutatások mind értékesítési, mind pedig pénzügyi szempontból. Előbbinél a statisztikai kombinációs lehetőségek tárháza végtelen, viszont jelenleg csak Státusztstatisztika, Termékstatisztika és az Értékesítők eredménystatisztikái láthatóak. Ezek havi bontásban jelennek meg, amit megváltoztathatunk, ezáltal segítve a havi értékesítési eredménykimutatások megírását. A pénzügyi statisztika is hasonló szerepet tölt be, ebben az esetben viszont egy összesített képet kap a felhasználó a bevételek alakulásáról.

Az alkalmazás egyszerű és letisztult megjelenítésével elősegíti az effektívebb információkeringést. Főként a PrimeNg angular UI könyvtárnak köszönhetően sikerült megvalósítani ezt a kifejezetten modern és felhasználóbarát felületet.

A legnagyobb segítséget a program íráskor a Google Firebase jelentette, melynek hála az alkalmazás rendelkezik saját adatbázissal, tárhellyel és autentikációval. Ezen kívül egy hosting

szolgáltatást is biztosít, melynek segítségével online telepített formában is elérhető a FOX-CRM számára, amely a következő címen elérhető:

<https://fox-crm-511f9.web.app/login>

Irodalomjegyzék

Az alábbi hivatkozások utoljára 2021.11.17.-én lettek megnyitva

[1] Csicsman József, Új Calculus Bt. Budapest: „Ügyfélkapcsolat-kezelés, a CRM rendszer segítségével”,
<http://www.inf.u-szeged.hu/~csicsman/calculus/CRM/CRM.pdf>

[2] Deltek Maconomy: ERP rendszerek fogalma és típusai,
<https://vallalatiranyitasi-rendszer.hu/crm-rendszerek-tipusai/>

[3] ChatCompose: Értékesítési folyamatok,
<https://www.chatcompose.com/hu/process.html>

[4] Angular hivatalos dokumentációja,
<https://angular.io/docs>

[5] Angular Materials könyvtár használati segédlet,
<https://material.angular.io/components/categories>

[6] PrimeNG könyvtár használati segédlet,
<https://primefaces.org/primeng/showcase>

[7] Firebase hivatalos dokumentációja,
<https://firebase.google.com/docs/web/setup>

[8] Jánki Zoltán Richárd, Kokrehel Gracijan, Szabó Zoltán: Webfejlesztési keretrendszerek gyakorlat, (SZTE TTIK)
<https://github.com/jankiz/Web-development-frameworks>

Nyilatkozat

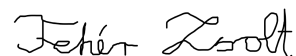
Alulírott Fehér Zsolt gazdaságinformatika BSc szakos hallgató, kijelentem, hogy a dolgozatomat a Szegedi Tudományegyetem, Informatikai Intézet Számítógépes Optimalizálás Tanszékén készítettem, gazdaságinformatika BSc diploma megszerzése érdekében.

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 a Szegedi Tudományegyetem Informatikai Intézet könyvtárában, a helyben olvasható könyvek között helyezik el.

2021. november. 29.

Aláírás



Köszönetnyilvánítás

Disztingvált módon szeretném megköszönni mindazoknak a közreműködését és zabolátlan segítségét, akik tanácsukkal és szaktudásukkal hozzájárultak a szakdolgozatom sikeres elkészítéséhez. Külön köszönet illeti témavezetőmet, Csicsman József tanár urat, akinek interdiszciplináris szakértelme és útmutatása nélkül ez nem jöhetett volna létre, illetve természetesen családomat, akik támogattak a tanulmányaim alatt.

Hatalmas köszönet továbbá barátaimnak (*U.w.U.*), akik az évek során mellettem álltak és töretlenül támogattak. Ők azok, akikre mindig, minden körülmények között számíthattam és akik miatt az egyetemen töltött éveim maradnak életem legszebb emlékei.

Elektronikus melléklet

A forráskód megtalálható az alábbi GitHub linken: <https://github.com/Zsolf/fox-crm>

- GitHub felhasználó: **Zsolf**
- Repository név: **fox-crm**

A forráskód fő ága a **main** branch, ahol a **fox-crm** mappában találjuk az **src** főkönyvtárat, amely a következő almappákból áll:

- **app** (Forráskódok)
- **assets** (Képfájlok)
- **environments** (Környezeti változók)

Az **app** mappában, avagy a forrásfájlokat tartalmazó könyvtárban találhatóak a következő almappák:

- **login** (Autentikációs oldalak)
- **pages** (Az alkalmazás többi oldala, modulokra bontva)
- **services** (Servicek)
- **shared** (Adatmodellek, Menübár, Authguard és Pipe-ok)

A gyökérkönyvtárban továbbá helyet kapott a szakdolgozatom pdf példánya:

Szakdolgozat_Feher_Zsolt.pdf