FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Návrh architektúry univerzálného licenčného servera
Student:	Richard Baláž
Vedoucí:	Ing. Marek Suchánek
Studijní program:	Informatika
Obor / specializace:	Webové a softwarové inženýrství, zaměření Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2022/2023

Pokyny pro vypracování

Licencování softwaru je netriviální záležitostí z pohledu efektivního řešení ověření licencí a správy jejich parametrů. Licenční servery umožňují efektivní správu licencí, ale jsou často vytvořeny pro specifický produkt nebo mají limitované strategie při práci licencemi. Cílem této práce je navrhnout univerzální a snadno rozšiřitelný licenční server:

- Analyzujte problematiku licencování software, ověřování a správy licencí.
- Proveďte rešerši existujících licenčních serverů, jejich technologií a porovnání. Prozkoumejte také dostupná open-source řešení.
- Navrhněte architekturu licenčního serveru pro správu a ověřování licencí s důrazem na bezpečnost i rozšiřitelnost a přizpůsobitelnost. Jako součást návrhu vytvořte specifikaci REST API.
- Implementujte prototyp řešení dle návrhu za použití frameworku ASP.NET Core a Entity Framework.
- Zhodnoťte přínosy vlastního řešení z pohledu bezpečnosti, rozšiřitelnosti i udržitelnosti. Stručně vlastní řešení porovnejte s existujícími z rešerše.

Elektronicky schválil/a Ing. Michal Valenta, Ph.D. dne 26. října 2021 v Praze.

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalárska práca

Návrh architektúry univerzálného licenčného servera

Richard Baláž

Katedra softwarového inženýrství
Vedúci práce: Ing. Marek Suchánek

3. apríla 2022

Poďakovanie

Ďakujem môjmu vedúcemu práce Ing. Marek Suchánek za užitočné pripomienky, ochotu a usmernenie pri písaní bakalárskej práce.
Taktiež ďakujem Bohu, svojej rodine a priateľom za ich neustálu morálnu podporu.

Prehlásenie

Prehlasujem, že som predloženú prácu vypracoval(a) samostatne a že som uviedol(uviedla) všetky informačné zdroje v súlade s Metodickým pokynom o etickej príprave vysokoškolských záverečných prác.

Beriem na vedomie, že sa na moju prácu vzťahujú práva a povinnosti vyplývajúce zo zákona č. 121/2000 Sb., autorského zákona, v znení neskorších predpisov, a skutočnosť, že České vysoké učení technické v Praze má právo na uzavrenie licenčnej zmluvy o použití tejto práce ako školského diela podľa § 60 odst. 1 autorského zákona.

V Prahe 3. apríla 2022 . . . . . .

Odkaz na túto prácu

Baláž, Richard. *Návrh architektúry univerzálneho licenčného servera.* Bakalárska práca. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2022.

Abstrakt

Táto bakalárska práca sa zaoberá návrhom a implementáciou licenčného servera s dôrazom na jeho univerzálnosť, bezpečnosť a prispôsobiteľnosť. Práca popisuje problematiku licencovania softvéru, vysvetluje princípy licenčných politík a oprávňovania použitia softvéru modelom licenčného server a klienta. Nosnou časťou práce je analýza požiadaviek, návrh a následná implementácia prototypu licenčného servera pre správu licencií a vydávanie aktivačných súborov vo frameworkoch *ASP.Net Core* a *Entity Framework*. V poslednej časti práce je prototyp licenčného servera zhodnotený z hľadiska bezpečnosti a prispôsobiteľnosti a následne je porovananý s existujúcimi riešeniami z rešerše.

Klíčová slova licencie, licenčný server, REST API, ASP.Net Core, Entity Framework

Abstract

This bachelor thesis deals with the design and implementation of a license server with emphasis on its versatility, security and adaptability. The thesis describes the theory of software licensing, explains the principles of licensing policies and authorizing the use of software by the license server and client model. The main part of the thesis is requirements analysis, design and subsequent implementation of a license server prototype for license management and activation files issuing using the frameworks *ASP.Net Core* and *Entity Framework*. In the last part of the thesis, the prototype of the license server is reviewed in terms of security, adaptability and is compared with existing solutions from the research afterwards.

Keywords license, license server, REST API, ASP.Net Core, Entity Framework

Obsah

Zoznam obrázkov

Zoznam tabuliek

Úvod

V počiatkoch informačnej éri, v 50-tych až 60-tych rokoch 20. storočia, boli softvérové programi písané väčšinou akademikmi a korporátnymi výskumníkmi a vydávali sa vrámci *public-domain*. Nástupom operačných systémov a internetu, softvérový priemysel začal narastať a v roku 1974 bol v USA prijatý zákon, ktorý rozhodol, že počítačový program je duševným vlastníctvom autora a môže byť copyrightovaný [3]. Tento zákon naplno otvoril dvere monetizácii softvéru a jeho licencovaniu. Výrobcovia softvéru sa s narastajúcimi prípadmi neoprávneného používania softvérov začínali zaujímať o riešenia, ktoré by ich ochránili. Úlohou licenčného klienta je dôveryhodne potvrdiť nainštalovanému softvéru oprávnenosť jeho používania.

Táto práca sa sústredí na analýzu, návrh a implementáciu takéhoto licenčného servera. V prvej kapitole sa snaží popísať problematiku licencovania softvéru, kategorizovať licenčné politiky, uviesť niekoľko spôsobov aktivácie softvéru a vysvetliť, prečo sa konvenčný softvér nedá ochrániť stopercentne. Následne je popísaný model fungovania licenčného servera s licenčným klientom. V druhej kapitole sú opísané funkcionality existujúcich, či už open-source alebo proprietárnych licenčných systémov. V tretej kapitole sú analyzované požiadavky, ktoré budú na navrhovaný prototyp licenčného servera kladené vrátane ich prioritizácie. Súčasťou tejto kapitoly sú aj prípady použitia a aj doménový model, ktorý spája pojmy do súvislosti. V štvrtej kapitole je opis návrhu prototypu licenčného servera a jeho architektúry. V tejto kapitole je navrhnutá aj REST API špecifikácia serveru a opis použitých technológii. Piata kapitola popisuje prototyp z implementačného hľadiska, najme pomocou diagramu tried. V šiestej kapitole je prototyp licenčného servera zhodnotený z hladiska bezpečnosti a prispôsobiteľnosti. V poslednej časti kapitoly je prototyp licenčného servera porovnaný s existujúcimi riešeniami z rešerše.

Kapitola 1

Licencovanie softvéru

1.1 ISV a softvérová licencia

Nezávislý dodávateľ softvéru (ISV) je spoločnosť alebo jednotlivec, ktorý vyvíja a predáva softvér, ktorý je spotrebovávaný koncovími používateľmi. Inými slovami, nezávislý dodávateľ softvéru je spoločnosť, ktorej jednou z hlavných podnikateľských činností je aj distribúcia softvéru.

Softvérová licencia je dohoda medzi vlastníkom majetkového práva (ISV), ktorý poskytuje softvérový produkt (alebo prípadne zdrojový kód) a jeho koncovým používateľom. Licenčná zmluva na softvér (tiež známa aj ako *EULA*) je právny dokument, ktorý stanovuje podmienky, za ktorých je koncový používateľ legálne oprávnený používať konkrétny softvér. Licenčná zmluva zvyčajne obsahuje ustanovenia týkajúce sa spôsobu použitia softvéru, ako je počet povolených inštalácií alebo akékoľvek obmedzenia úpravy a redistribúcie softvéru. [4]

1.1.1 FOSS softvérové licencovanie

FOSS (free and open-source software) licencie poskytujú koncovému používateľovi práva, ktoré zahŕňajú úpravu, opätovné použitie a distribúciu zdrojového kódu softvéru alebo softvérového produktu. Tento typ licencovania s otvoreným zdrojovým kódom poskytuje používateľovi oprávnenie upravovať funkcie softvéru a slobodne používať zdrojový kód. [5]

Príkladmi takéhoto licencovania sú licencie *GNU General Public Licence, BSD license alebo MIT license.*

1.1.2 Proprietárne softvérové licencovanie

Proprietárne licencie neposkytujú žiadne oprávnenie na úpravu alebo opätovné použitie kódu a bežne poskytujú softvér iba vo forme spustiteľného binárneho kódu. Proprietárna softvérová licencia často ob-

sahuje podmienky, ktoré zakazujú „reverzné inžinierstvo“ binárneho kódu s úmyslom získať zdrojový kód koncovým používateľom. [5]

Príkladmi takéhoto licencovania sú licencie *EULA Microsoft Windows, Adobe Software License Agreement alebo Rockstar Games End User License Agreement.*

1.2 Licenčná politka softvérov

Licenčná politika sa dá definovať ako podmienky, za ktorých ISV licencuje svoje softvéry zákazníkom. Zvyčajne sa odvíjajú od princípov a cieľov podnikateľského plánu ISV. Vhodným nastavením licenčnej politiky sa ISV snaží maximalizovať a stabilizovať zisk z predaja softvéru. Licenciu softvéru môže ponúkať v rôznych variantách, ktoré sa líšia rozsahom, časovou platnosťou, prenostiteľnosťou a funkciami. V nasledujúcich častiach sú stručne popísané rôzne spôsoby licencovania podľa rozsahu a časovej platnosti podľa [6] .

1.2.1 Licencovania podľa rozsahu

- **Node-locking** – kedysi najpopulárnejšia forma licencovania sa zakladá na tom, že softvér je licencovaný pre konkrétny počítač. Tento licenčný model je zvyčajne použitý vo výpočtovo náročných aplikáciách, kedy sú pracovné stanice zvyčajne určené práve na použitie daného licencovaného softvéru. Pre zabezpečenie sa *aktivačný kľúč* zvyčajne viaže na sériové čísla hardvéru konkrétneho počítača. Nevýhodou je, že *aktivačný kľúč* je nepoužiteľný po výmene pokazeného hardvéru. Po vydaní nového *aktivačného kľúču* nie je možné aby si bol ISV istý, že predchádzajúci *aktivačný kľúč* nie je používaný naďalej, teda či nedochádza k porušovaniu licenčnej zmluvy.

- **User-based licensing** – model, ktorý sa zakladá na tom, že softvér je licencovaný konkrétnemu používatelovi. Tento model je užitočný pre softvéry, ktoré sú závislé od používateľa, napr. e-mailový klient alebo bankové aplikácie – teda aplikácie, pri ktorých je „požičanie“ identifikácie používateľa niekomu inému v rozpore so samotnou podstatou aplikácie.

- **Floating licences** – takzvané „plávajúce“ licencie je model, pri ktorom je licencia vrámci spoločnosti požičiavaná koncovými používateľmi iba počas používania licencovaného softvéru. To umožňuje používanie licencií viacými koncovými používateľmi pri tej podmienke, že nikdy nie je prekročený limit maximálneho počtu bežiacich instancií softvéru (teda počtu licencií, ktoré spoločnosť vlastní). Tento licenčným model sa stal prevažujúcim najmä vďaka efektívnemu využitiu počtu zakúpených licencií.

1.2.2 Licencovania podľa časovej platnosti

- **Večné licencie** – najtradičnejší model licencovania, kedy výrobca softvéru (ISV) predá právo na používanie softvéru na večnú dobu jednorázovým poplatkom. Koncový používateľ má nárok na opravné aktualizácie po dobu záruky. Na ďalšie softvérové aktualizácie koncový používateľ zvyčajne nárok nemá. V poslednej dobe sa od tohto modelu licencovania ustupuje a nahradzuje sa *modelom predplatného* z dôvodu vyššej efektivity monetizácie. [7]

- **Model predplatného** – model, kedy konečný používateľ si opakovane kupuje právo na používanie softvéru (typicky na mesiac alebo na rok). Výhodou pre konečného používateľa je nižšia počiatočná cena a možnosť kedykoľvek si predplatné zastaviť. Pre výrobcu softvéru (ISV) tento model predstavuje spôsob, akým si zaistiť pravidelný príjem a monetizovať softvérové aktualizácie.

- **SaaS** – najflexibilnejší model, kedy si konečný používateľ platí za použitie aplikácie cez webové rozhranie. Odpadáva pritom nutnosť inštalácie na strane používateľa a nutnosť aktivácie. Tento model je účtovaný tzv. consumption-based modelom – napr. spotrebovaným CPU časom servera výrobcu (ISV), kde softvér vykonáva výpočty.

1.3 Zabezpečenie proprietárneho licencovania

Po predaji povolenia používánia softvéru (licenčná zmluva) je v záujme ISV nutné zabezpečiť, aby softvér mohol byť použiteľný iba v tom rozsahu ako je dohodnuté v licenčnej zmluve. Toto zabezpečenie chráni ISV pred ušlými ziskami neoprávneným používaním softvéru.

1.3.1 Softvérová aktivácia

Softvérová aktivácia je technológia určená na verifikáciu toho, že softvér sa používa legálne.

Existuje nikoľko spôsobov implementácie tejto technológie:

- **Aktivačný kľúč** – softvéry vyžadujú aktivačný kľúč. Zvyčajne sa softvér nedajú nainštalovať alebo spustiť, kým nie je zadaný platný aktivačný kľúč. Aktivačný kľúč je zvyčajne reťazec znakov, ktorý obsahuje kombináciu písmen a číslic.

  - *Offline aktivácia* – aktivačný kľúč môžu byť súčasťou produktu v krabici alebo môže byť odoslaný e-mailom po objednávke. Implementačne to býva riešené funkciou, ktorá overuje určité matematické vlastnosti aktivačného kľúču, ktoré musia byť splnené.

  - *Online aktivácia* – metóda aktivácie softvéru, ktorá zahŕňa databázu aktivačných kódov vydaných ISV (zvyčajne súčasť licenčného servera). Aktivačné kľúče sa dostávajú ku koncovým používateľ obdobným spôsobom ako u *offline aktivácii*. Koncoví používatelia aktivačné kľúče manuálne zadajú do softvéru alebo inštalátora. Softvér kontaktuje databázu ISV a zistí, či aktivačný kľúč existuje a je validný.

- **Hardvérové kľúče** – elektronické hardvérové kľúče sú mechanizmy určené na ochranu pred neautorizovaným použitím softvéru. Najčastejšie dodávaný vo forme *dongle* - malé elektronické zariadenie s *USB* pripojením, ktoré sa pripája do počítača, kde sa má softvér používať. Dongle obsahuje údaje o licencií, ktoré sú prečítané licencovaným softvérom.

- **Node-bound softvér** – po zakúpení node-locked licencie sa na strane ISV začne proces zostavovania unikátneho inštalátora softvéru, ktorý je naviazaný na hardvérové prostriedky počítača, na ktorom bude softvér použitý. Koncovému používateľovi stačí softvér nainštalovať týmto inštalátorom a nie je potrebná žiadna ďalšia aktivácia. [8]

- **Cloud and ID-based** – licencovaný softvér vyžaduje na fungovanie pripojenie k licenčnému serveru nejakou identitou (zvyčajne používateľské meno a heslo). Licencovaný softvér si dokáže odkomunikovať a verifikovať oprávnenosť používania softvéru s licenčným serverom. Licenčný server periodicky vysiela *heartbeat* signál, ktorý „udržuje licencovaný softvér pri živote“.

Zmienené spôsoby implementácie majú väčšinou vždy nejakú svoju nevýhodu:

Offline aktivačný kľúč môže byť vygenerovaný kýmkoľvek pomocou malej utilitou, tzv. *keygen-om*.

Hardvérové kľúče spôsobujú komplikácie - je ich nutné fyzicky distribuovať, môžu sa pokaziť, nie je možné upraviť ich obsah na diaľku (verziu licencované softéru, funkcie,...). Taktiež neumožňujú používateľsky prívetivý floatingový model licencii.

Node-bound distribúcia softvéru požaduje zostavenie individuálneho inštalačného balíčka pre každého jedného používateľa, čo je výpočtovo náročné pri softvéroch s veľkým množstvom používateľov.

Cloud-based aktivácia vyžaduje z definície neustále pripojenie k licenčnému serveru, či už lokálneho alebo vzdialeného. Výpadok tohto serveru spôsobí aj priamy výpadok licencovaného softvéru používateľom.

V praxi sa vyššie uvedené technológie zvyčajne kombinujú, aby sa dosiahla vysoká bezpečnosť a požadovaná flexibilita.

Príklad: hardvérový kľúč môže byť použitý k autentifikácii na licenčný server v cloude, ktorý odovzdá licencovanému softvéru aktivačný klúč platný na nejakú dobu. [6]

1.3.2 Aktivačný súbor

Aktivačný súbor je súbor obsahujúci informácie o licencii, jej parametroch a ďalších potrebných informácii daného licenčného modelu. Dôležitou súčasťou tohto súboru je aj digitálny podpis, ktorým sa zaisťuje integrita toho, že tento súbor bol vydaný licenčným serverom a nikto ho neupravoval. Pojem aktivačný súbor je rozšírením aktivačného kľúču. Zvyčajne obsahuje nasledujúce informácie:

- *druh licenčného modelu* – napríklad: večná licencia, floating licencia, licencia s obmedzenou časovou platnosťou, a pod.
- *identita licencovaného softvéru* - zvyčajne to býva kódový názov a verzia softvéru
- *zoznam funkcií (features)* – súčasti softvéru, ktoré boli zakúpené a nie sú štandardne k dispozícii
- *identita určeného počítača* – typicky sériové čísla hardvérových komponentov, ktoré musia sedieť s tými na použitom hardvéri
- *dátum a čas expirácie* – od kedy nie je možné licenčný súbor považovať za platný
- *digitálny podpis* – kontrolný súčet všetkých obsiahnutých informácií, zašifrovaný tajným kľúčom pomocou asymetrického šifrovania

1.3.3 Generovanie aktivačného súboru a jeho overovanie

Pokiaľ ide o licencovanie softvéru, algoritmy generovania aktivačných súborov a overovacích algoritmov, ktoré si ISV zvolí, môžu byť dôležitým aspektom licencovania. Po napadnutí algoritmu generovania už ISV nemôže dôverovať žiadnym predtým vygenerovaným aktivačným súborom, vrátane tých, ktoré patria legitímnym koncovým používateľom.

„Crackovanie“ softvéru je akt priamej úpravy kódu softvéru tak, aby sa úplne obišiel jej licenčný systém. Všetky aplikácie nainštalované na zariadení koncových používateľov sú náchylné na tento typ útoku a z princípu neexistuje pred nimi stopercentná ochrana.

Softvérový „keygen“ je oveľa zlovestnejší typ útoku. Keygen je forma softvéru, často samostatná utilita alebo webová stránka, ktorá generuje platné aktivačné kľúče/súbory.

ISV má nejaký typ keygenu, ktorý uchováva v tajnosti, či už je to algoritmus na strane licenčného servera alebo samostatný softvér na generovanie.

Pri použití starých a zraniteľných algoritmoch na generovanie a verifikáciu aktivačného kľúča/súboru, môže útočník byť schopný dosiahnuť generovanie platných, aj keď nelegitímnych aktivačných kľúčov/súborov, za predpokladu, že vynaloží určité úsilie na reverzné inžinierstvo algoritmu.

1.3.3.1 Parciálne overenie kľúča (PKV)

Parciálne overenie kľúča (PKV) (podľa [9]) je algoritmus aktivačného kľúča, ktorý rozdeľuje aktivačný kľúč na viacero „podkľúčov". pri každej novej verzii licencovaného softvéru skontroluje algoritmus overenia aktivačného kľúča inú podmnožinu podkľúčov licencie.

Verifikačný algoritmus nikdy netestuje úplný aktivačný kľúč, testuje iba podmnožinu podkľúčov.

Nevýhody PKV sú:

- postupom času unikne algoritmus generovania aktivačného kľúča
- je nutné udržiavať čiernu listinu uniknutých/nelegitímnych kľúčov
- pri znalosti určitého množstva legitímnych kľúčov je možné algoritmus odvodiť
- je náročné vložiť informácie o licencii do kľúča

1.3.3.2 Overenie aktivačného súboru digitálnym podpisom

Overenie aktivačného súboru digitálnym podpisom je moderným spôsob ako overiť, že aktivačný súbor bol vydaný dôveryhodným zdrojom, využíva sa pritom asymetrický kryptografický algorimus - napríklad *RSA* alebo *eliptické krivky*. Spôsob, akým tieto asymetrické kryptografické algoritmy fungujú, je, že majú súkromný kľúč a verejný kľúč. Vezmú sa informácie o licencii (*payload*) a vytvorí sa ich digitálny podpis pomocou súkromného kľúča, ktorý je možné overiť pomocou verejného kľúča. Overenie je v podstate kontrola pravosti: „boli tieto údaje podpísané súkromným kľúčom?"

Ako už názov napovedá, súkromný kľúč je tajomstvom ISV (tj. nikdy sa nezdieľa) a verejný kľúč je vložený do kódu licenčného klienta licencovaného softvéru.

Výhodou tejto metódy je, že pri použití napr. *Ed25519* alebo *RSA-2048* je dnes výpočtovo nezvládnutelné šifru prelomiť, teda napísať keygen je prakticky nemožné.

Avšak, *crack* nie je *keygen*, takže licencovanie softvéru vždy predstavuje riziko, že môže byť obídené úpravou binárneho kódu. [9]

1.3.4 Ochrana softvéru pred crackingom

Cracking je upravovanie alebo odstraňovani ochranných mechanizmov softvéru za účelom nelegálneho používania softvéru.

Licencovaný softvér, ktorý používa nejakú formu softvérovej aktivácie, obsahuje logiku, podľa ktorej sa program rozhoduje, či je jeho používanie oprávnené.

Cieľom akejkoľvek ochrany softvéru nie je chrániť softvér, ale namiesto toho odradiť potenciálneho crackera zložitým a ťažko zrozumiteľným binárnym kódom softvéru.

Pre akýkoľvek licencovaný softvér, ktorý je chránený, predpokladáme, že cracker má absolútnu kontrolu nad hardvérovými a softvérovými komponentami systému, na ktorom je softvér spustený. Hardvér môže byť emulovaný alebo navrhnutý na mieru. Sieť môže byť izolovaná LAN. To znamená, že existuje málo komponentov systému, ak vôbec nejaké, ktoré dokáže softvér predpokladať, že sú dôveryhodné. To ale neznamená, že písanie softvérovej ochrany je zbytočné. Skôr to znamená, že je nutné si stanoviť reálne ciele pre ochranu softvéru. V nasledujúcej časti sú uvedené niektoré príklady implementačných techník ochrany softvéru pred crackingom, podľa [10].

1.3.4.1 Príklady implementačných techník

Detekcia modifikácie softvéru – detekcia, či boli binárne súbory softvéru modifikované, je v podstate problémom detekcie chýb, kedy sa používa algoritmus kontrolného súčtu, ako napríklad CRC32, MD5 alebo SHA1. Tieto algoritmy je možné použiť na generovanie podpisu pre ľubovoľný blok kódu alebo dát. Tento podpis je potom možné skontrolovať za behu, aby sa zistilo, či došlo k nejakej modifikácie.

Obfuskácia kódu – účelom obfuskácie kódu je vytvoriť udržiavateľný zdrojový kód, ktorý beží rýchlosťou blízkou originálu, ale je ťažké ho pochopiť, keď je reprezentovaný v strojovom jazyku. Toto sa dá dosiahnuť zjavným zvýšením zložitosti alebo falošnými vyjadreniami konštánt, dátových typov a podmienených výrazov používaných v algoritmoch. Medzi ďalšie techniky patrí stieranie hraníc medzi funkciami preložením kódu dvoch alebo viacerých funkcií do viacúčelovej funkcie pomocou vlastných virtuálnych strojov alebo emulátorov. Ďalej je možné využívať reprezentáciu funkcie v bajtovom kóde, vytvárať nové vlákna alebo procesy na vykonávanie triviálnych alebo irelevantných úloh.

Používanie ukazovateľov na funkcie – adresa funkcie bude vždy viditeľná v pamäti pred jej volaním, avšak uložením obfuskovanej verzie ukazovateľa funkcie, disassemblery a nástroje na analýzu odkazov nedokážu rozpoznať uložený ukazovateľ ako adresu kódu.

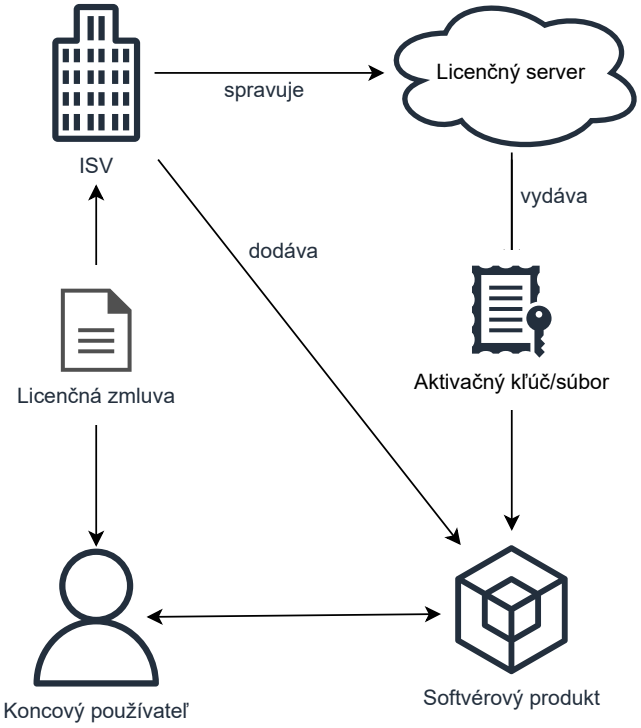
1.4 Licenčný server a klient

Licenčný server je centralizovaný počítačový softvér pre správu licencií poskytujúci dôveryhodné informácie pre klientsky softvér o jeho oprávnení byť používaný. Zodpovednosťou licenčného servera je dohliadať na počet kópii softvéru, ktoré môžu byť oprávnene používané na základe licenčnej zmluvy.

Rozdelenie licenčných serverov podľa umiestnenia:

- **Vzdialený licenčný server** – autorita, ktorá vydáva aktivačné súbory je prevádzkovaná výrobcom softvéru (ISV) a riziko nedostupnosti znáša ISV. Výhodou je úplna kontrola nad aktualizáciami.
- **Lokálny licenčný server** – autorita, ktorá vydáva aktivačné súbory je prevádzkovaná samotným koncovým používateľom. Výhodou je, že počítače koncového používateľa nemusia mať prístup na internet. Nevýhodou sú vyššie prevádzkové náklady.

Licenčný klient je softvér, ktorý riadi oprávnenie spúšťať a používať podliehajúci licencovaný softvér na základe *aktivačného súboru*, ktorý je odkomunikovaný s licenčným serverom. Na obrázku 1.1 je znázornené umiestnenie licenčného servera v doméne licencovania.



Obr. 1.1: Diagram licencovania

1.4.1 Správa licencií

Úlohou *vzdialeného licenčného servera* okrem vydávania aktivačných súborov je aj udržiavanie databázy licencií a ich spravovanie. Typické úkony správy licencie zahŕňajú priradenie novej licencie koncovému používateľovi, úprava dostupných funkcií pre licenciu alebo prípadne odňatie licencie (*CRUD*). Licenčné servery typicky poskytujú okrem administračného panelu aj nejakú formu automatizovanej správy - napríklad prepojenie na objednávkový systém. [1]

1.4.2 Všeobecné cieľe licenčného servera

Všeobecné ciele licenčného servera sa môžu na prvý pohľad zdať jednoduché. Keďže sa však správa licencií stáva dôležitou službou v sieti, takéto ciele môžu byť pomerne zložité. Licenčný server musí byť veľmi robustní – záložné licenčné servery sú nevyhnutné. Ak licenčný server zlyhá, používatelia nedokážu spustiť zakúpený softvér - je nutná vysoká dostupnosť aj redundacia siete.

Licenčný server musí byť dostatočne flexibilný aby umožňoval licencovanie softvéru rôznymi licenčnými politikami. Licenčný server musí rozumieť aj nepriamočiarym konceptom, ako napríklad floating licencie.

Hoci koncept licencovania je dobre akceptovaný, nie je až taká jednotnosť v tom, čo je „používanie“ licencie. Napríklad niektoré softvérové licencie umožňujú jednému používateľovi spustiť viac ako jednu „kópiu“ softvéru na rovnakom s rovnakou identitou používateľa. Správna odpoveď sa líši v závislosti od licencovaného softvéru. Správca licencií musí umožniť rôzne interpretácie toho, čo je „použitie“ licencie. [6]

1.4.3 Typická komunikácia klient-server

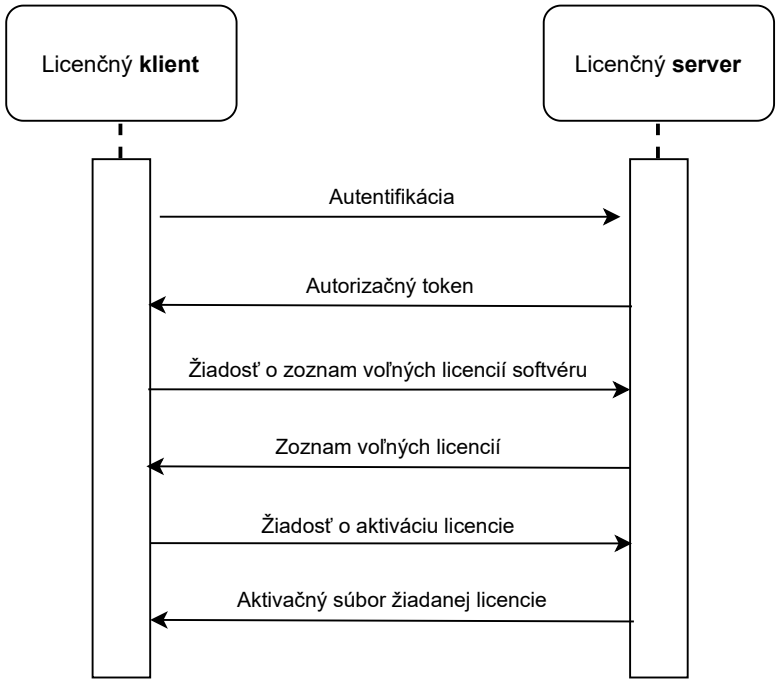
Pri komunikácií klienta s licenčným serverom, je zvyčajne nutné, aby licenčný klient bol pre server identifikovateľný na základe nejakej identity. V prvom kroku komunikácie, zvyčajne prebieha autentifikáca identity. Autentifikáciu môže vykonať licenčný server alebo externý autentifikačný server. Licenčný klient po autentifikácii disponuje autorizačným tokenom, ktorý je použitý v ďalšej komunikácii s licečným serverom.

Licečný klient zozbiera informácie o podliehajúcom licencovom softvéri – napríklad kód produktu, verzia a pošle požiadavky na server, ktorou sa dožaduje o zoznam voľných licencií, ktoré má autentifikovaná identita (koncový používateľ) k dispozícii. Licenčný klient, či už na základe výberu používateľa alebo nejakého algoritmu požiada o aktiváciu niektorej z voľných licencií. Do žiadosti o aktiváciu sa zvyčajne pridáva identita hardvéru počítača.

Licenčný server sa rozhodne licenciu aktivovať – označí ju za aktívnu a vydá digitálne podpísaný *aktivačný súbor* pre licenčného klienta. Nasleduje

proces, v ktorom sa licenčný klient rozhoduje, či *aktivačný súbor* je dostatočná informácia na autorizovanie používania podliehajúceho softvéru.

Komunikácia licenčného klienta a licenčného servera je zachytená na diagrame 1.2. [2]



Obr. 1.2: Diagram komunikácie licenčený klient - licenčný server

Kapitola 2

Rešerš existujúcich licenčných serverov

2.1 Výber existujúcich riešení pre rešerš

Pri výbere z celej rady dostupných riešení na licencovanie softvéru sa táto práca zameriava na tie najrozšírenejšie najrozšírenejšie.

FlexNet Publisher je narozšírenejšie komerčné riešenie pre licencovanie softvéru používaný významnými softvérovými spoločnosťami ako napríklad *Adobe*, *SolidWorks* alebo *Hexagon AB*. [11, 12, 13]

Riešenie *License4j* bolo vybrané na základe dostupnej rozsiahlej dokumentácie, jednoduchému použitiu a nízkej obstarávacej cene.

Naopak, *Open License Manager* spolu s *License3j* patria medzi najpopulárnejšie repozitáre na GitHub na tému "license management" s open-source licenciou.

V nasledujúcich častiach sú popísané niektoré vybrané open-source a komerčné riešenia na licencovanie softvéru podľa [14], [15], [1] a [2]. Ako bude možné vidieť v nasledujúcich častiach, komerčné licencovacie nástroj zvyčajne predstavujú reálne použitelné riešenia. Naopak, riešenie pre licencovanie sofvéru je mierne kontroverznou témou v open-source komunite a existujúce riešenia neimplementujú dostatočnú funkcionalitu.

2.2 Open License Manager

Open Licence Manager (ďalej ako *OLM*) je open-source systém vhodný na licencovanie a chránenie pred neautorizovanými kópiami pre relatívne malé projekty. Systém je licencovaný pod *BSD 3* a umožňuje voľné použitie aj v komečnej sfére. Systém taktiež obsahuje licenčného klienta `licensecc`, ktorý sa integruje do softvérového produktu a zabezpečuje overovanie aktivačných kľúčov. Aktiváciou pri tomto systéme sa rozumie prečítanie a valido-

vanie aktivačného súboru licenčným klientom. Systém neimplementuje žiadny plnohodnotný licenčný server a neexistuje žiadna komunikácia licenčného klienta s vonkajším svetom. Systém nepodporuje žiadnu správu vydaných licencií. Namiesto toho systém na serverovej strane pozostáva z utility `lccgen`, ktorá generuje plnohodnotné digitálne podpísané aktivačné súbory.

OLM podporuje vydávanie večných licencií, licenci s expiračnou dobou a umožňuje pre každú licenciu definovať aj zoznam softvérových features alebo špecifický payload pre softvérový produkt.

OLM sa primárne zameriava na licenčnú politku node-locked licencií, avšak podporuje aj tzn. trial licencie – aktivačné súbory s vymedzenou dobou platnosti na vyskúšanie softvérového produktu koncovým používateľom bez node-lockingu.

V neposlednom rade je v tomto systéme prítomný aj `lcc-inspector` – debugovací nástroj určený na použitie u koncového používateľa pre identifikáciu rôznych licenčných problémov alebo na zistenie hardvérových reťazcov, na ktoré je možné licenciu naviazať.

2.3 Licence3j

Licence3j je ďalšou open-source knižnicou na vydávanie aktivačných súborov a ich čítanie podobná *OLM*. Nejedná sa o plnohodnotný systém s licenčným serverom schopný vyhovieť požiadavkam licenčného klienta. Knižnica je vydaná pod *Apache 2.0* licenciou. Systém umožňuje cez *REPL utilitu* vytvárať a modifikovať aktivačné súbory licencie. Licencia v *Licence3j* je definovaná veľmi všeobecne a rozšíriteľne pomocou množiny viacero `feature`, kde jedna `features` pozostáva z mena, typu hodnoty a hodnoty samotnej. Teda nepodporuje priamo funkcionality ako node-locking, časovo obmedezné licencie. Avšak táto funkcionalita môže byť doplnená vlastným dodefinovaním toho, čo niektoré `feature` znamenajú a doimplementovať ich v licenčnom kliente. Licenčné súbory vydávané týmto systémom sú digitálne podpísané.

2.4 Licence4j

License4j je komplexné proprietárne riešenie licenčného servera a licenčného klienta, primárne určené pre softvérové produkty napísané v *Jave*. Ekosystém zahŕňa:

`License Manager` – licenčný manažér pre správu licencií,

`Auto License Generation and Activation Server` – server pre vydávanie aktivačných súborov a online aktiváciu,

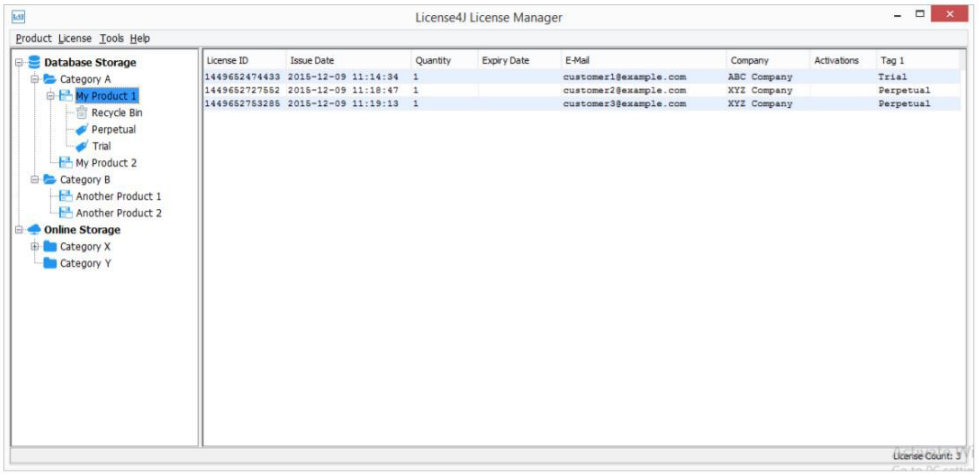
`License Activation and Validation Proxy Server` – aktivačný server určený do lokálnej sieti koncového používateľa.

Licence4j podporuje rôzne druhy modelov licencovania: večné licencie, časovo obmedzené licencie, floating licencie, node-locked licencie. Zároveň podporuje niekoľko možných druhov aktivácie aktivačným kľúčom/súborom:

- **Aktivačný súbor** – zašifrovaný text definovaných vlastností licencie,
- **Floating aktivačný súbor** – aktivačný súbor rozšírený o funkcionalitu požičiavania licencií z `License4J Floating License Server`.
- **Jednoduchý aktivačný kľúč** – aktivačný kľúč generovaný reverzibilným algoritmom. Tento kľúč nie je kryptograficky bezpečný. Môže obsahovať ID hardvéru ak je licencia node-locked.
- **Kryptograficky zabezpečený aktivačný kľúč** – kryptograficky bezpečný kľúč generovaný algoritmom eliptických kriviek. Oproti jednoduchému aktivačnému kľúču je tento kľúč dlhší.
- **Online jednoduchý floating aktivačný kľúč** – kľúč určený pre funkcionalitu požičiavania licencií z `License4J Floating License Server`.

Systém na správu licencií `License Manager` je jednoduchý softvér s GUI umožňujúci spravovanie licencií a produktov. `License Manager` umožňuje spravovať generované licencií vrámci produktov.

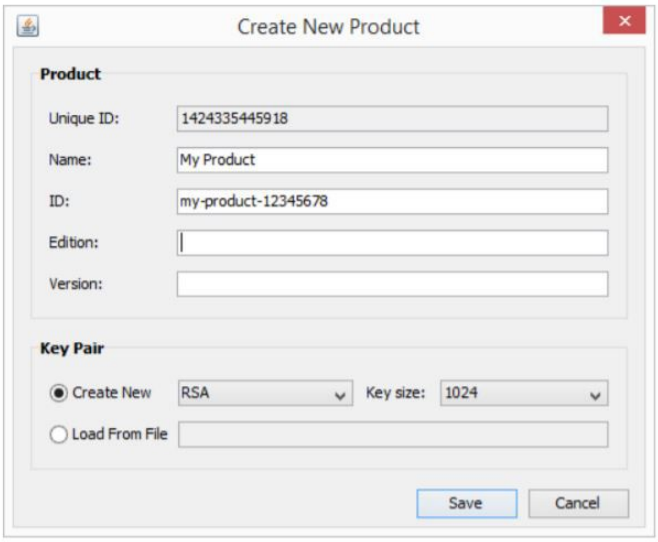
Pohľad na hlavné okno `License Manager` je zachytený na obrázku 2.1.



Obr. 2.1: Hlavné okno systému License Manager [1]

Pri vytvorení produktu (softvéru, ktorý má podliehať licencovaniu) sa vygeneruje pár verejného a súkromného kľúča, ktorý už nie je možné nikdy zmeniť. Tieto kľúče sú použité na validáciu a generovanie aktivačných súborov.

Každý produkt má svoje unikátne ID, svoje meno, verziu a edíciu. Sprievodca vytvárania produktu je zachytený na obrázku 2.2.



Obr. 2.2: Sprievodca vytvárania produktu v License4j [1]

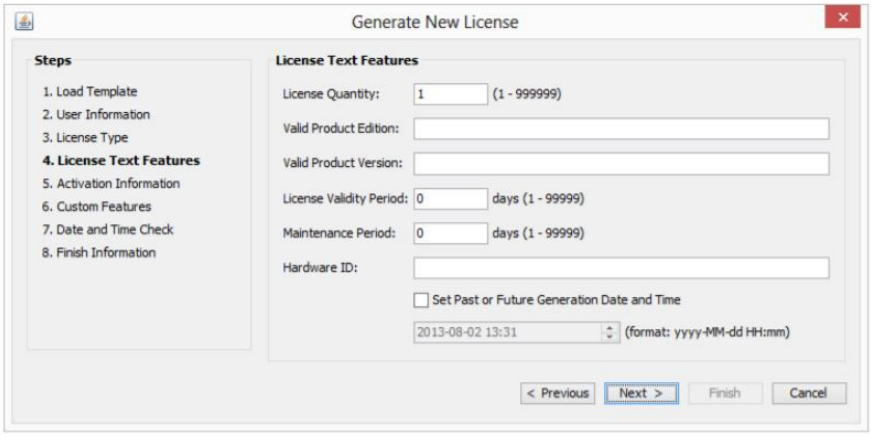
Pri vytváraní licencií sa používa podobný sprievodca ako pri produkte. Sprievodca vytvárania licencií umožňuje vybrať šablónu, vyplniť informácie o zakazníkovi, zvoliť aktivačnú metódu, zadať počet použití licencie, povolené verzie a edície licencovaného softvéru, expiračnú dobu licencie a hardvérové ID, ak má byť node-locked. V sprievodcovi je možné navyše zadefinovať produktovú funkcionalitu - množinu *features*. *Licence4j* umožňuje vynutíť porovnávanie lokálneho času licenčného klienta s internetovým časom (ak sa nezhodujú, licencia nebude licenčným klientom považovaná za platnú). Sprievodca vytvárania licencie je zachytený na obrázku 2.3.

Licence4j implementuje aj mechanizmus deaktivácie licencie. Deaktivácia licencie umožňuje koncovému používateľovi prenášať licenciu na iný počítač alebo iného používateľa. Túto funkcionalitu je možné povoliť v správe licencií a nastaviť maximálny počet deaktivácii na obmedzenie – toto obmedzenie sa snaží zamedziť nadmernému použitiu licencie.

Manažment aktivácií umožňuje správcovi licencií manuálne aktivovať licencie, zobraziť informácie o aktivácii a deaktivovať licenciu.

2.4.1 Integrácia licenčného klienta

Knižnica `License4J-Library-Runtime.jar` je knižnica licenčného klienta, ktorá musí byť použítá v licencovanom softéri. Nevýhodou *Licence4j* je, že licenčný klient je na teraz napísaný iba v jazyku *Java* a neposkytuje bindings pre iné programovacie jazyky.



Obr. 2.3: Sprievodca vytvárania licencie v License4j [1]

Použitie licenčného klienta je veľmi priamočiare – pri aktivačnej metóde aktivačného súboru/kľúču stačí použiť metódu `LicenseValidator.validate` nasledovne:

```
LicenseValidator.validate(
    licenseString, // aktivačný kľúč/súbor (vyplní používateľ)
    publickey, // verejný kľúč tohto produktu
    productID, // ID tohto produktu
    productEdition, // edícia tohto produktu
    currentProductVersion, // verzia tohto produktu
    currentDate, // aktuálny čas
    currentProductReleaseDate // čas vydania tohto produktu
);
```

Metóda vracia objekt typu `License`, ktorý obsahuje niekoľko informácii vrátane statusu aktivačného kľúču/súboru – či je aktivačný kľúč/súbor platný.

2.5 FlexNet Publisher

FlexNet Publisher (predtým FLEXlm) je štandardom v komečnej sfére pre softvérové licencovanie založenom na digitálnych podpisoch. *FlexNet Publisher* je používaný tisíckami ISV, ktorým poskytuje jednoduché riešenie pre licencovanie a ochranu sofvéru. Systém podporuje rôzne modely licencovania, umožňuje monitorovanie využitia licencií, ponúka pokročilú ochranu proti neoprávneným kópiam a poskytuje vysokú dostupnosť.

Definícia licencie vo *FlexNet Publisher* sa odvíja od zvoleného licenčného modulu, ale zvyčajne zahŕňa nasledujúce informácie:

- aké softvérové funkcie je možné použiť – môžu byť samostatne licencované,
- aké verzie softvéru je možné použiť,
- platformy, na ktorých je softvér možné použiť,
- obdobie (čas expirácie licencie), počas ktorého je možné softvér používať.

Aktivačný súbor s licenciou môže byť uložený:

- *V aktivačnom súbore* – textový súbor s príponou `.lic`, ktorého obsah je chránený digitálnymi podpismi, ktoré sú overené prostredníctvom licenčných komponentov FlexNet Publisher.
- *V dôveryhodnom úložisku* – bezpečné miesto, ktorého obsah je šifrovaný. Licencie v dôveryhodnom úložisku môžu čítať iba licenčné komponenty FlexNet Publisher.

Licencovaný softvér, ktorý implementuje licenčného klienta tohto systému (ďalej ako *FlexEnabled aplikácia*) môže získať aktivačný súbor buď priamo z licenčného súboru alebo z lokálneho dôveryhodného úložiska na lokálnom počítači. Niektoré licenčné modely, ktoré sú založené na online aktivácii, poskytujú licencie, ktoré sú centrálne spravované ISV licenčným serverom. *FlexEnabled aplikácie* sú spojené s týmto licenčným serverom cez sieť TCP/IP a požadujú od neho aktivačné súbory.

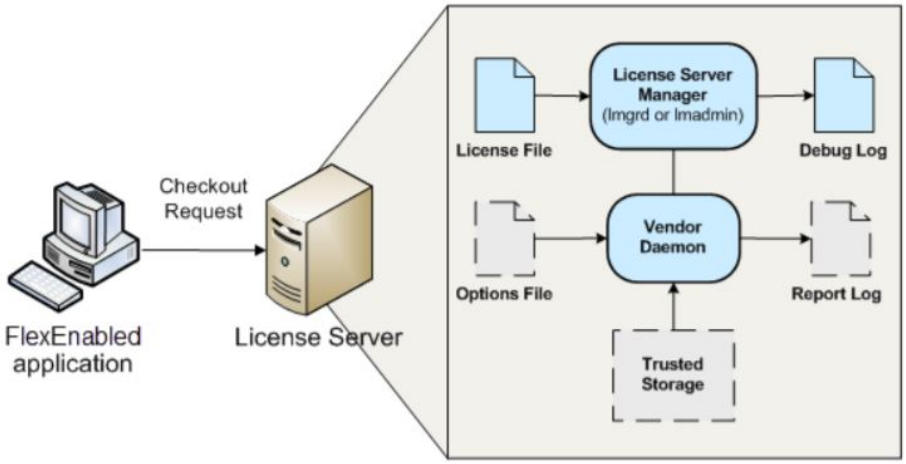
FlexNet Publisher pomedzi iné licenčné modely, podporuje aj floating licencie, ktoré nie sú viazané na konkrétny počítač ale stanovujú maximálny počet používateľov, ktorí môžu naraz softvér používať.

Základné komponenty systému *FlexNet Publisher* sú zachytené na obrázku 2.4.

2.5.1 Licenčný súbor

Licenčný súbor v systéme *FlexNet Publisher* obsahuje potrebné informácie ohľadom manažovania licencií pre FlexEnabled aplikáciu licenčným serverom. Jeho obsah tvorí: názov licenčných serverov a ich identifikátora `hostids`, perióda heartbeat signálu, informácie o ISV a zoznam softvérových funkcií s digitálnym podpisom a iné. V tomto súbore sa definuje aj licenčný model. Príklad obsahu licenčného súboru je uvedený na 2.1.

Tento príklad umožní licenčnému serveru s názvom `my_server` s hostid `17007ea8` ponúkať 10 licencií pre každý softvérovú funkcionalitu `f1` a `f2` pre akéhokoľvek používateľa na sieti.



Obr. 2.4: Základné komponenty systému FlexNet Publisher [2]

```
SERVER my_server 17007ea8 1700
VENDOR sampled
FEATURE f1 sampled 1.000 31-dec-2020 10 SIGN="<...>"
FEATURE f2 sampled 1.000 31-dec-2020 10 SIGN="<...>"
```

Kód 2.1: Registrácie implementácii rozhrania

2.5.2 Podporované licenčné modely

FlexNet Publisher podporuje veľa licenčných modelov. Medzi ne patria aj nasledujúce:

- *Floating licencie* – floating licencia znamená, že ktokoľvek v sieti môže používať FlexEnabled aplikáciu až do limitu uvedeného v licenčný súbor. Floating licencie nemajú žiadne hostid na jednotlivých `FEATURE` záznamoch.

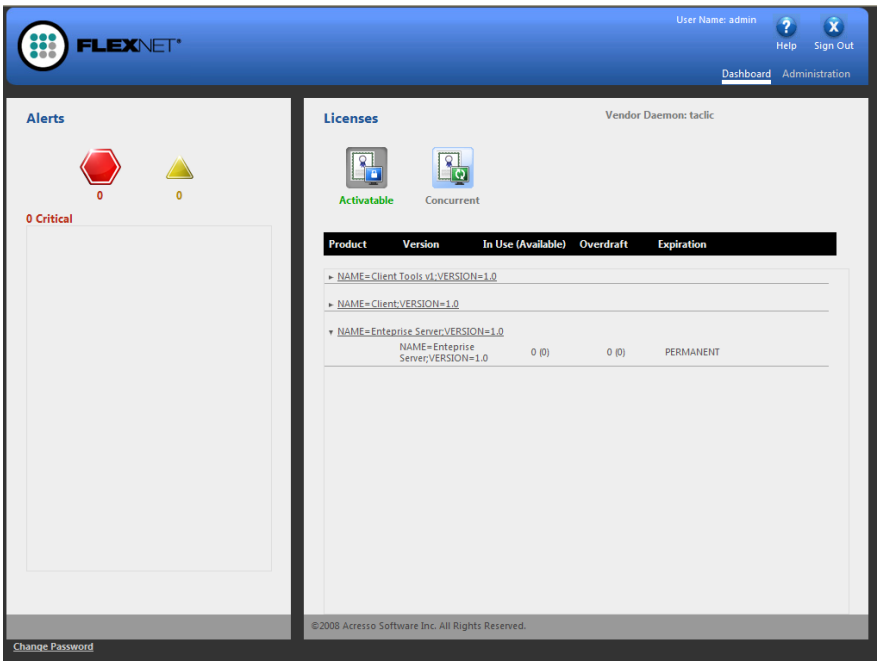
- *Node-locked licencie* – node-locked licencie znamená, že FlexEnabled aplikácia môže byť použitá iba na jednom konkrétnom počítači. Node-locked licencia má vyplnené hostid vo `FEATURE` záznamoch, ktoré identifikujú špecifický počítač.

- Mixované floating a node-locked licencie – uvedeným kľúčového slova `uncounted` do `FEATURE` záznamu sa licencia dá použiť kdekoľvek na sieti.

- Dongle licencie – licencie používajú namiesto hostid iný identifikátor – identifikátor hardvérového dongla. Dongle je vyrábaný ISV, ktorý ho následne distribuuje koncovým používateľom.

2.5.3 Správa licencií

Webová aplikáca `lmadmin`, ktorá je súčasťou *FlexNet Publisher* umožňuje spravovať licencie - generovať ich, aktivovať, deaktivovať, mazať. Zároveň je možné vidieť stav a využiteľnosť licencií. Náhľad webovej aplikácie je na obrázku 2.5.

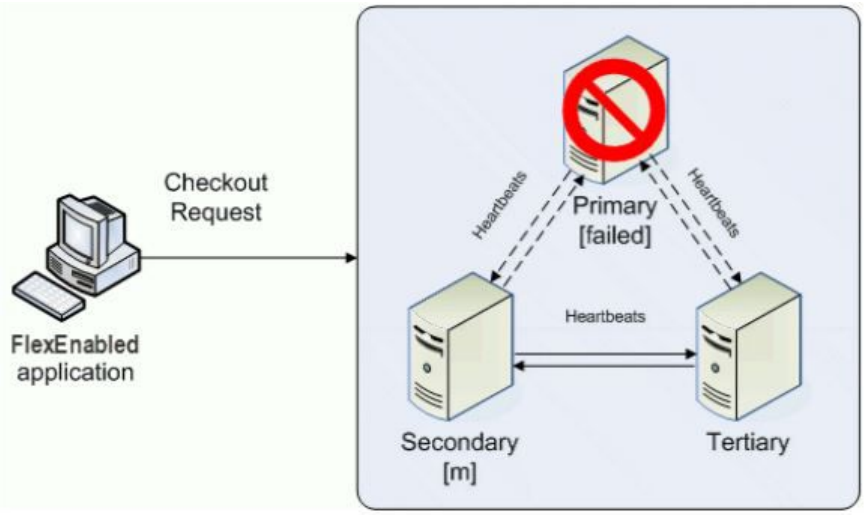


Obr. 2.5: Webová aplikácia lmadmin [1]

2.5.4 Zabezpečenie dostupnosti

FlexNet Publisher umožňuje nakonfigurovanie takzvanej *Three-server redundancie* – pomocou možnosti redundancie troch licenčných serverov fungujú všetky tri licenčné servery tak, že tvoria triádu. Licenčné servery si navzájom posielajú pravidelné správy, aby sa uistili, že existujú aspoň dva servery beh a komunikácia. Ak hlavný server zlyhá, hlavným serverom sa stáva iný licenčný server. Táto technológia umožuje fungovanie služby licencovania

aj v prípade zlyhania licenčného servera. Situácia, kedy hlavný server zlyhá a jeho úlohu preberie sekundárny server je zachytená na obrázku 2.6.



Obr. 2.6: Diagram prepojenia licenčných serverov FlexNet Publisher a výpadku primárneho servera [2]

2.6 Porovnanie existujúcich riešení

Vyššie uvedené open-source riešenia *OLM* a *License3j* sú oproti komerčným licenciam obmedzené, pretože postrádajú možnosť správy licencií a online aktivácie licencií, a preto ich nasadenie v praxi je limitované iba na prípady offline aktivácii bez možnosti spravovania vydaných licencií (môže byť použiteľné pri malom množstve zákazníkov). Na druhú stranu, obe vyššie spomenuté komerčené riešenia *Licence4j* a *FlexNet Publisher* majú plnohodnotný webový manažér licencií určený na ich správu. Ďalej vidíme, že komerčné riešenia poznajú veľa druhov konceptov licencovania, a teda sú univerzálne. Výhodou *License4j* je, že umožňuje si vybrať ISV z viacero druhov aktivačných kľúčov alebo súborov. Jeho nevýhodou je absencia licenčných klientov v rôznych programovacích jazykov. Veľkou výhodou *FlexNet Publisher* je podpora dôveryhodného úložiska (využitie TPM) a zabezpečenie redundacie technológiou *Three-server redundancy*. Rovnako ako pri *Licence4j*, aj *FlexNet Publisher* nepodporuje consuption-based licencovanie. Prehľad podporovaných funkcionalít týchto existujúcich riešení je zhrnutý v tabuľke 2.1.

Funkcionalita	**OLM**	**License3j**	**Licence4j**	**FlexNet Publisher**
Licenčná zmluva	BSD 3	Apache 2.0	proprietárna	proprietárna
Manažment licencií	-	-	✓	✓
Online aktivácia licencie	-	-	✓	✓
Offline aktivácia licencie	✓	✓	✓	✓
Večné licencie	✓	✓	✓	✓
Časovo obmedzené licencie	✓	-	✓	✓
Consumption-based licencie	-	-	-	-
Node-locked licencie	-	✓	✓	✓
Floating licencie	-	-	✓	✓
Porpora features	✓	✓	✓	✓
Licenčný klient	✓	✓	✓	✓
Digitálne podpisovanie licencií	✓	✓	✓	✓

Tabuľka 2.1: Prehľad podporovaných funkcionalít existujúcich riešení

Kapitola 3

Analýza

3.1 Cieľová skupina

Cieľom práce je navrhnúť licenčný server tak, aby bol univerzálnym, teda aby podporoval viaceré licenčné modely, ktoré sa v praxi najčastejšie vyskytujú. Konečným používateľom licenčného servera je ISV, ktorý bude používať tento server na správu licencií a vydávanie aktivačných súborov pre jeho vlastné softvérové produkty.

Od konečného používateľa sa očakáva, že tento systém napojí do svojho CRM a komunikáciu s licenčným serverom bude riadiť svojim licenčným klientom.

Pre účely analýzy a návrhu v tejto práci definujeme nasledujúce role používateľov, ktoré s licenčným serverom interagujú:

- *Vlastník licencie* - je každá osoba (alebo spoločnosť), ktorá si od ISV zakúpila licenciu na nejaký softvérový produkt. Tento používateľ je vedený v systéme CRM vydatateľa softvéru pod unikátnym identifikátorom.

- *Koncový používateľ* - je osoba patriaca pod vlastníka licencie (napr. zamestnanec), ktorý licenciu používa. Tento používateľ by mal byť vedený licenčným serverom ako počítač s vydaným aktivačným súborom.

- *Administrátor licencií ISV* - je osoba ktorá zastupuje ISV (zamestnanec ISV) a je poverená manažmentom licencií. Táto osoba má právo vytvárať, upravovať a mazať licencie na licenčnom serveri.

3.2 Požiadavky

V nasledujúcej časti sú popísané požiadavky, ktoré sú na navrhovaný licenčný server kladené. Požiadavky vychádzajú najmä z existujúcich a chýbajúcich

funkcionalít preskúmaných open-source licenčných riešení z rešeršnej kapitoly. Pri definovaní požiadavok sa prihliadalo aj na vymoženosti komerčných riešení, ktoré poskytli pohľad na problematiku licencovania ich rokmi vývoja a spätných väzieb v reálnom nasadení. V poslednom rade, požiadavky bezpečnosti a rozšíriteľnosti vychádzajú aj zo zadania bakalárskej práce.

3.2.1 Prioritizácia metódou MoSCoW

Metóda **MoSCoW** je jednoduchý spôsob ako zoradiť požiadavky podľa ich priority a tým pochopili ktoré požiadavky sú dôležité k nasadeniu a ktoré nie sú až tak podstatné z hľadiska použiteľnosti. Ku každej požiadavke môže byť priradená jedna z nasledujúcich priorít [16]:

- *Must have* – požiadavky s kritickou prioritou, ktoré sú esenciálne pre základnú funkčnosť. Tieto požiadavky pri nasadení musia byť implementované.
- *Should have* – požiadavky, ktoré sú dôležité, ale nie sú kritické.
- *Could have* – požiadavky, ktoré môžu uľahčiť prácu ale nie sú až tak dôležité.
- *Won't have* – požiadavky, ktoré nie sú implementované, ale môžu byť témou budúceho vývoja.

3.2.2 Klasifikácia podľa FURPS

FURPS je akronym pre *Functionality, Usability, Reliability, Performance a Supportability* a je to typ klasifikácie používaný pre funkčné, ale aj nefunkčných požiadavky softvérových systémov. Požiadavky klasifikuje do nasledujúcich kategórii [17]:

- *Functionality* – Požiadavka opisuje hlavnú funkcionalitu a schopnosť softvéru podporovať biznis procesy.
- *Usability* – Požiadavka sa zameriava na zjednodušenie použiteľnosti softvéru používateľom.
- *Reliability* – Požiadavka, ktorá má za účelom zlepšiť spolahlivosť softvéru a znížiť *MTBF (mean time between failures)*.
- *Performance* – Požiadavka, ktorá určuje rýchlosť odozvy softvéru a aj zaťaženie systému.
- *Supportability* – Požiadavka, ktorá má za účel zlepšiť udržateľnosť aplikácie, zjednodušiť ďalší vývoj alebo testovateľnosť.

3.2.3 Funkčné požiadavky

Funkčnými požiadavkami licenčného servera sa rozumie zoznam funkcionality (features), ktoré server musí pre koncového používateľa podporovať.

- **F1 – Server bude umožňovať definovať licencovaný produkt.**
  Priorita: Must have
  Systém bude udržiavať databázu licencovaných produktov a umožní ich vytváranie – vygeneruje dvojicu verejného a súkromného kľúča určeného na podpisovanie aktivačných súborov a jeho validáciu.

- **F2 – Server bude umožňovať vytváranie a čítanie licencií.**
  Priorita: Must have
  Systém bude manažovať vlastnú databázu licencií, bude ponúkať rozhranie pre vytváranie, čítanie licencií. Zároveň licencia sa bude dať naviazať na niekoľko konkrétnych produktov s konkrétnou verziou a edíciou a povolenou funkcionalitou.

- **F3 – Server bude umožnovať upravovanie licencií**
  Priorita: Should have
  Systém bude okrem vytváranie a mazanie podporovať aj funkciu upravenia existujúcej licencie.

- **F4 – Licencia bude môcť byť priradená vlastníkovi licencie.**
  Priorita: Must have
  Systém bude môcť byť napojený na externú databázu vlastníkov licencie cez autorizačný server OpenID. Licencia bude môcť byť priradená vlastníkovi licencie vzťahom "používateľ vlastní licenciu".

- **F5 – Licencia bude môcť byť zdieľaná viacerými vlastníkmi licencií**
  Priorita: Could have
  Systém umožní licenciu priradiť nie len jednému, ale aj viacerým vlastníkom naraz.

- **F6 – Server bude poskytovať rozhranie pre aktiváciu licencií.**
  Priorita: Must have
  Systém bude ponúkať funkcionalitu aktivácie licencie – vydanie aktivačného súboru pre licenčného klienta a vyhodnocovať oprávnenie aktivácie buď na základe licenčného kľúča alebo autorizačným tokenom a identifikátorom licencie. Podporované metódy budú online aktivácia a aj offline aktivácia.

- **F7 – Server bude vydávať aktivačný súbor uzamknutý iba na jeden konkrétny počítač.**
  Priorita: Must have
  Aktivačný súbor je vydaný a platný iba pre jeden konkrétny počítač. Tento súbor by nemal byť použiteľný na inom počítači.

- **F8 – Server umožní vydať aktivačný súbor iba na niekoľko konkrétnych počítačov uvedených v licencii**
  Priorita: Could have
  Server nedovolí vydať aktivačný súbor na ľubovolný počítač. Počítač, pre ktorý sa aktivačný súbor má vydať musí byť v zozname povolených počítačov licencie.

- **F9 – Server bude podporovať licenčnú politiku večných licencií.**
  Priorita: Must have
  Licencia, ktorá bude mať označenie večnej licencie, musí môcť byť aktivovateľná vždy (pokiaľ je voľná).

- **F10 – Server bude podporovať licenčnú politiku dočasných licencií.**
  Priorita: Should have
  Licencia, ktorá má uvedenú platnosť je aktivovateľná iba v čase jej platnosti.

- **F11 – Server bude podporovať licenčnú politiku floating licencií.**
  Priorita: Could have
  Licencia, ktorá je označná ako floating licencia je vydávaná iba na krátku dobu a predlžovaná (obsadzovaná) opakovanou "heartbeat" požiadavkou počas jej používania.

- **F12 – Server bude podporovať licenčnú politiku consumption-based licencií.**
  Priorita: Could have
  Pokiaľ licencia má uvedenú konzumovateľnú jednotku (napr. čas CPU), tak každým použitím licencie sa táto konzumovateľná jednotka znižuje o spotrebované množstvo. Ak konzumovateľná jednotka dosiahne nulovú hodnotu, licencia už nie je ďalej aktivovateľná.

- **F13 – Licencia bude upgradovateľná na vyššiu verziu softvéru.**
  Priorita: Won't have
  Server umožní jednoducho a automatizovane pridať najnovšiu verziu softvéru do povolených produktov licencie.

- **F14 – Server bude môcť byť napojený na objednávkový systém.**
  Priorita: Won't have
  Napojenie objednávkového systému na licenčný server zautomatizuje vytváranie licencií a ich správy.
- **F15 – Server bude poskytovať GUI na správu licencií.**
  Priorita: Won't have
  Okrem API bude licenčný server poskytovať aj grafické užívateľské rozhranie pre správu licencií a ich monitoring.

3.2.4 Nefunkčné požiadavky

Nefunkčnými požiadavkami licenčného servera sa rozumejú požiadavky, ktoré popisujú ako by mal systém fungovať vrámci jeho použitia a môže definovať obmedzujúce podmienky dizajnu, ktoré by mal spĺňať.

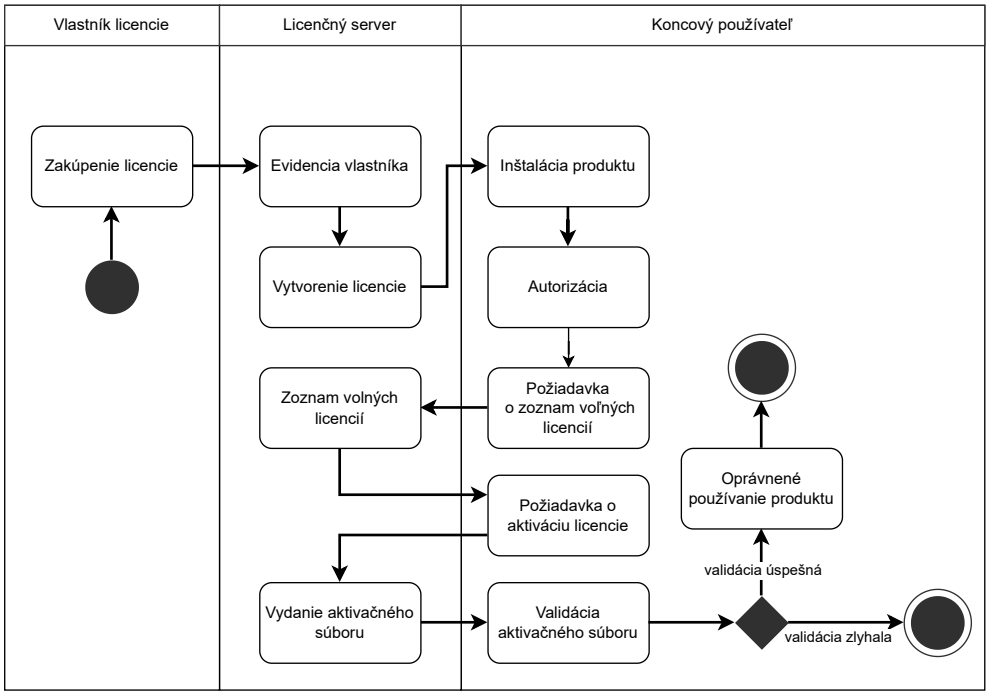
- **N1 – Server bude môcť byť použitý na rôznych platformách.**
  Kategória: Functionality
  Systém musí podporovať operačné systémy *Windows Server* a *Linux*.
- **N2 – Server bude rozhranie ponúkať cez HTTPS protokol.**
  Kategória: Functionality
- **N3 – Rozhranie musí byť dodržiavať REST API koncepty.**
  Kategória: Usability
- **N4 – Server bude musieť podporovať SQL databázu odolnú voči výpadkom prúdu.**
  Kategória: Reliability
- **N5 – Server bude môcť pôsobiť ako záložný server pre iný server.**
  Kategória: Reliability
- **N6 – Server by mal zvládať požiadavky paralelne.**
  Kategória: Performance
- **N7 – Kritické funkcie servera musia byť zabezpečené.**
  Kategória: Functionality
  Vytváranie licencie a vydávanie aktivačného súboru musia byť z hľadiska bezpečnosti implementované správne, aby sa predošlo prelomeniu.
- **N8 – Funkcionalita licenčného servera musí byť jednoducho rozšíriteľná.**
  Kategória: Supportability
- **N9 – Licenčný server musí podporovať nasadenie cez Docker.**
  Kategória: Supportability

3.3 Typický pracovný postup

Typický pracovný postup s licenčným serverom je nasledovný: ako prvé si osoba si zakúpi licenciu k produktu, čím sa stáva vlastníkom licencie. Administrátor ISV ho zaeviduje v jeho CRM a vytvorí mu v licenčnom serveri licenciu podľa objednávky. Koncový používateľ si produkt nainštaluje a prihlási sa do jeho licenčného klienta produktu účtom vlastníka licencie, čím získa autorizačný token. Následne si z licenčného servera prostredníctvom licenčného klienta vypýta zoznam voľných licencií. Z týchto licencií si jednu vyberie a zašle na licenčný server požiadavku o jej aktiváciu. Licenčný server prevedie aktiváciu a vydá koncovému používateľovi aktivačný súbor. Licenčný klient koncového používateľa tento súbor prečíta, overí jeho vlastnosť a povolí používanie nainštalovaného produktu.

Alternatívne, po zakúpení licencie môže byť koncovému používateľovi predaný licenčný kľúč – tajný identifikátor licencie. Server umožňuje aktiváciu licencie pomocou tohto kľúču aj bez autorizačného tokenu.

Na obrázku 3.1 je znázornený typický pracovný postup aktivácie autorizačným tokenom opísaný v prvom odseku pomocou activity diagramu.



Obr. 3.1: Diagram typického pracovného postupu aktivácie autorizačným tokenom

3.4 Prípady použitia

Prípady použitia detailnejšie popisujú rôzne situácie a scenáre toho, ako okolie interaguje so softvérom.

Prípady použitia sú podrobnejšie zachytené v ďalšej časti. Diagram prípadov použitia z pohľadu administrátora licencií ISV je znázornený na obrázku 3.2 a z pohľadu koncového používateľa je znázornený na obrázku 3.3.

- **UC1 – Vytvorenie definície produktu**
  Účastníci: administrátor licencií ISV
  Podmienky: Produkt s daným menom neexistuje
  Scenár:

  1. Administrátor licencií ISV zadá meno nového produktu, vyberie algoritmus asymetrického šifrovania a vyberie povolené licenčné modely pre nový produkt.
  2. Server vygeneruje verejný a súkromný kľúč a definíciu produktu uloží.

- **UC2 – Vytvorenie licencie**
  Účastníci: administrátor licencií ISV
  Scenár:

  1. Administrátor licencií ISV vyberie produkty, ku ktorým sa licencia vzťahuje, zadá jej meno, vyberie licenčný model a zakliká povolené spôsoby aktivácie.
  2. Administrátor licencií ISV vyberie povolené verzie produktov, vytvorí zoznam povolených funkcionalít (features), a zároveň vyplní detaily licenčného modelu, ako napríklad dĺžka platnosti licencie, prenositeľnosť, počet spustiteľných instancií a pod.
  3. Administrátor licencií ISV priradí licenciu konkrétnemu vlastníkovi licencie.
  4. Server vygeneruje licenčný kľúč a licenciu uloží.

- **UC3 – Upravenie licencie**
  Účastníci: administrátor licencií ISV
  Scenár:

  1. Administrátor licencií ISV si zo zoznamu licencii vyberie licenciu ktorú chce upraviť.
  2. Administrátor licencií upraví ľubovoľnú vlastnosť licencie a zmeny odošle.
  3. Server zmeny licencie uloží.

- **UC4 – Priradenie licencie vlastníkovi licencie**
  Účastníci: administrátor licencií ISV
  Scenár:

  1. Administrátor licencií ISV si zo zoznamu licencii vyberie licenciu ktorú chce priradiť.
  2. Administrátor licencií ISV zadá unikátny identifikátor vlastníka licencie
  3. Server pridá identifikátor medzi vlastníkov licencie a zmenu uloží.

- **UC5 – Požiadavka o zoznam voľných licencií**
  Účastníci: koncový používateľ
  Podmienky: Licenčný klient nemá aktivovanú licenciu.
  Scenár:

  1. Koncový používateľ sa prostredníctvom licenčného klienta autentifikuje ako vlastník licencie cez externý autorizačný server OpenID a spolu s autorizačným tokenom odošle na licenčný server požiadavku o zoznam takých licencií, ktoré mu patria a sú voľné.
  2. Server požiadavku spracuje, načíta voľné licencie používateľa a odošle ich koncovému používateľovi.

- **UC6 – Požiadavka o zoznam licencií**
  Účastníci: administrátor licencií ISV
  Podmienky: Licenčný klient nemá aktivovanú licenciu.
  Scenár:

  1. Administrátor licencií ISV odošle požiadavku o zoznam všetkých licencií, ktoré sú v systéme.
  2. Server požiadavku spracuje, načíta licencie a odošle ich.

- **UC7 – Vydanie aktivačného súboru**
  Podmienky: Koncový používateľ prostredníctvom licenčného klienta odoslal požiadavku o aktiváciu licencie. Licencia musí byť aktivovateľná.
  Scenár:

  1. Server overí, či je aktivácia možná. Ak je aktivácia možná, systém vypracuje aktivačný súbor a podpíše ho všetkými súkromnými validačnými kľúčmi daných produktov.
  2. Následne systém označí licenciu za aktivovanú a aktivačný súbor odošle prostredníctvom licenčného klienta koncovému používateľovi.
  3. Licenčný klient koncového používateľa aktivačný súbor overí verejným validačným kľúčom. V prípade, že overenie prebehne v poriadku, licencia je načítaná do pamäte a používanie softvéru je považované za oprávnené.

- **UC8 – Aktivácia licencie metódou online aktivácie autorizačným tokenom**
  Účastníci: koncový používateľ
  Podmienky: Licencia musí mať túto metódu aktivácie povolenú.
  Scenár:

  1. Koncový používateľ prostredníctvom licenčného klienta požiada o zoznam voľných licencií
  2. Koncový používateľ si zo zoznamu voľných licencií vyberie práve jednu licenciu.
  3. Koncový používateľ prostredníctvom licenčného klienta priamo požiada licenčný server o aktiváciu danej licencie identifikátorom licenčného kľúča a dodá aj autorizačný token (získaný od autorizačného servera).
  4. Server požiadavku spracuje a vydá aktivačný súbor.

- **UC9 – Aktivácia licencie metódou online aktivácie licenčným kľúčom**
  Účastníci: koncový používateľ
  Podmienky: Licencia musí mať túto metódu aktivácie povolenú.
  Scenár:

  1. Koncový používateľ zadá licenčný kľúč do licenčného klienta .
  2. Koncový používateľ prostredníctvom licenčného klienta priamo požiada licenčný server o aktiváciu licencie s identifikátorom tohto licenčného kľúča.
  3. Server požiadavku spracuje a vydá aktivačný súbor.

- **UC10 – Aktivácia licencie metódou offline aktivácie**
  Účastníci: koncový používateľ, administrátor licencií ISV
  Podmienky: Licencia musí mať túto metódu aktivácie povolenú.
  Scenár:

  1. Koncový používateľ prostredníctvom licenčného klienta vygeneruje požiadavkový súbor o licenciu, ktorý bude obsahovať informácie o použitom softvéri.
  2. Koncový používateľ odošle požiadavkový súbor administrátorovi licencií ISV, spolu aj s licenciou, ktorú si praje aktivovať.
  3. Administrátor licencií ISV zadá požiadavkový súbor do licenčného servera spolu s identifikátorom licencie, ktorý ho overí a vydá aktivačný súbor.
  4. Administrátor licencií ISV odošle koncovému používateľovi aktivačný súbor.

5. Koncový používateľ vloží aktivačný súbor do licenčného klienta, ktorý overí ho overí.

- **UC11 – Aktivácia večnej licencie**
  Podmienky: Koncový používateľ prostredníctvom licenčného klienta odoslal požiadavku o aktiváciu licencie. Licencia musí mať priradenú licenčnú politiku večnej licencie.
  Scenár:

  1. Server požiadavku spracuje, overí či je licencia voľná a aktivuje ju na dlhšiu dobu.

- **UC12 – Aktivácia dočasnej licencie**
  Podmienky: Koncový používateľ prostredníctvom licenčného klienta odoslal požiadavku o aktiváciu licencie. Licencia musí mať priradenú licenčnú politiku dočasnej licencie.
  Scenár:

  1. Server spracuje o aktiváciu licencie, overí či je licencia voľná, overí či je aktuálne licencia platná porovnaním aktuálneho času s vlastnosťou licencie a aktivuje ju na dobu neprekračujúcu platnosť licencie.

- **UC13 – Aktivácia floating licencie**
  Podmienky: Koncový používateľ prostredníctvom licenčného klienta odoslal požiadavku o aktiváciu licencie. Licencia musí mať priradenú licenčnú politiku floating licencie.
  Scenár:

  1. Server licenciu aktivuje podobne ako v UC11, ale iba na krátku dobu.

- **UC14 – Aktivácia consumption-based licencie**
  Podmienky: Koncový používateľ prostredníctvom licenčného klienta odoslal požiadavku o aktiváciu licencie. Licencia musí mať priradenú licenčnú politiku consumption-based licencie.
  Scenár:

  1. Server spracuje požiadavku o aktiváciu licencie, overí či je licencia voľná, overí či je licencia má dostatok konzumovateľnej jednotky, odráta z konzumovateľnej jednotky požadované množstvo a licenciu aktivuje aktivuje na veľmi krátku dobu.

- **UC15 – Nastavenie automatického upgradovania licencií**
  Účastníci: administrátor licencií ISV

Scenár:

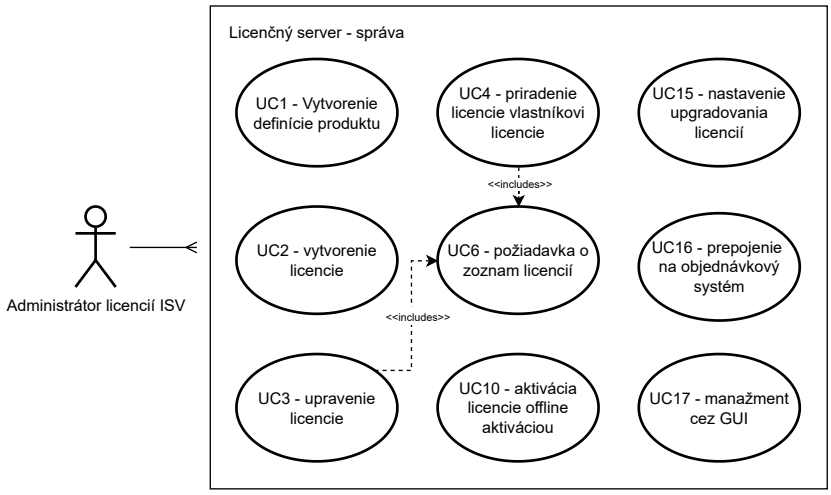
1. Administrátor licencií prepojí automatizačným skriptom jeho systém na zostavovanie produktu s licenčným serverom.
2. Server pri každom vydaní novej verzie produktu pridá verziu do vybraných licencií.

- **UC16 – Nastavenie prepojenia na objednávkový systém**
  Účastníci: administrátor licencií ISV
  Scenár:

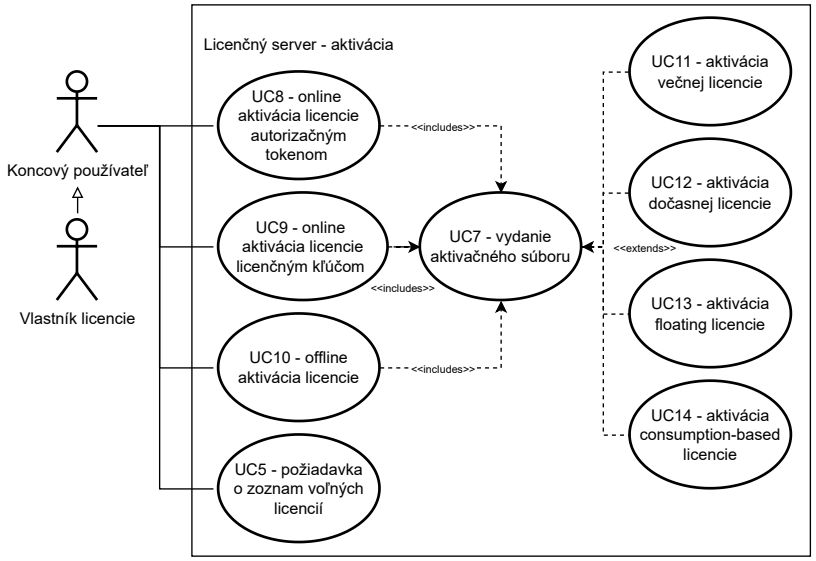
  1. Administrátor licencií ISV prepojí automatizačným skriptom jeho objednávkový systém s licenčným serverom.
  2. Server pri každej objednávke vytvorí príslušné licencie automaticky.

- **UC17 – Manažment servera cez GUI**
  Účastníci: administrátor licencií ISV
  Scenár:

  1. Administrátor licencií ISV požiada cez HTTP o grafické rozhranie licenčného servera.
  2. Server vráti grafické rozhranie v podobe HTML stránok a JS kódov.
  3. Administrátor licencií ISV pracuje s grafickým rozhraním a všetky požiadavky na server vykonáva cez toto rozhranie.



Obr. 3.2: Diagram prípadov užitia licenčného servera z pohľadu administrátora licencií ISV



Obr. 3.3: Diagram prípadov užitia licenčného servera z pohľadu koncového používateľa

3.4.1 Mapovanie FP na PP

Väzba funkčných požiadavok licenčného servera na prípady použitia je znázornená v tabuľke 3.1.

3.5 Doménový model

Doménový model je konceptovým model, ktorý dokumentuje jednotlivé objekty a vzťahy medzi nimi na vyššej úrovni.

Diagram doménového navrhovaný licenčný server je zobrazený na obrázku 3.4.

Pojem *licencia* predstavuje právo na používanie softvéru, ktoré si vlastník licencie od ISV zakúpil a koncový používateľ ju používa. Každá licencia má svoj atribút licenčného kľúču, ktorý predstavuje tajný reťazec pomocou ktorého sa dá licencia jednoznačne identifikovať.

Ku každej *licencií* sa vzťahuje niekoľko *licenčných obmedzení*, ktoré obmedzujú jej použitie. *Časová platnosť* licencie obmedzuje právo použia licencii iba na konkrétnu dobu. *Vlastníctvo* licencie stanovuje vlastníka licencie, ktorý má právo licenciu užívať (aktivovať). *Node-lock* licencie uzamyká aktivovateľnosť licencie iba na konkrétny počítač. *Povolený produkt* obmedzuje použitie licencie na konkrétny produkt s konkrétnou funkcionalitou. Takýto model obmedzení umožňuje obmedzenie rôzne kombinovať a povoľuje aj viacero obmedzení rovnakého druhu – napríklad viac obmedzení druhu *Povolený produkt* umožňuje licenciu použit na viacerých produktoch zároveň.

	F1 – definícia produktu	F2 – CR licencie	F3 – upravovanie licencie	F4 – priradenie licencie	F5 – zdieľaná licencia	F6 – aktivácia licencie	F7 – uzamknutie akt. súboru	F8 – uzamknutie licencie	F9 – večné licencie	F10 – dočasné licencie	F11 – floating licencie	F12 – consumption-based lic.	F13 – auto. upgrade licencie	F14 – objednávkový systém	F15 – GUI
UC1	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-
UC2	-	✓	-	-	-	-	-	✓	✓	✓	✓	✓	-	-	-
UC3	-	-	✓	-	-	-	-	-	-	-	-	-	-	-	-
UC4	-	-	-	✓	✓	-	-	-	-	-	-	-	-	-	-
UC5	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-
UC6	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-
UC7	-	-	-	-	-	✓	✓	-	✓	✓	✓	✓	-	-	-
UC8	-	-	-	-	-	✓	-	-	-	-	-	-	-	-	-
UC9	-	-	-	-	-	✓	-	-	-	-	-	-	-	-	-
UC10	-	-	-	-	-	✓	-	-	-	-	-	-	-	-	-
UC11	-	-	-	-	-	✓	-	-	✓	-	-	-	-	-	-
UC12	-	-	-	-	-	✓	-	-	-	✓	-	-	-	-	-
UC13	-	-	-	-	-	✓	-	-	-	-	✓	-	-	-	-
UC14	-	-	-	-	-	✓	-	-	-	-	-	✓	-	-	-
UC15	-	-	-	-	-	-	-	-	-	-	-	-	✓	-	-
UC16	-	-	-	-	-	-	-	-	-	-	-	-	-	✓	-
UC17	-	-	-	-	-	-	-	-	-	-	-	-	-	-	✓

Tabuľka 3.1: Mapovanie funkčných požiadavok na prípady použitia

Licencia môže mať niekoľko *zvláštnych licenčných politík* – vlastnosti licencie, na ktoré licenčný klient neprihliada a slúžia iba na strane servera. Obsah týchto *politík* môže byť časom menení používaním licencie, narozdiel od *licenčných obmedzení*.

Ku každej *licencii* sa vzťahujú aj *spôsoby akým môže byť aktivovaná* (online aktivácia licenčným kľúčom alebo autorizačným tokenom, offline aktivácia).

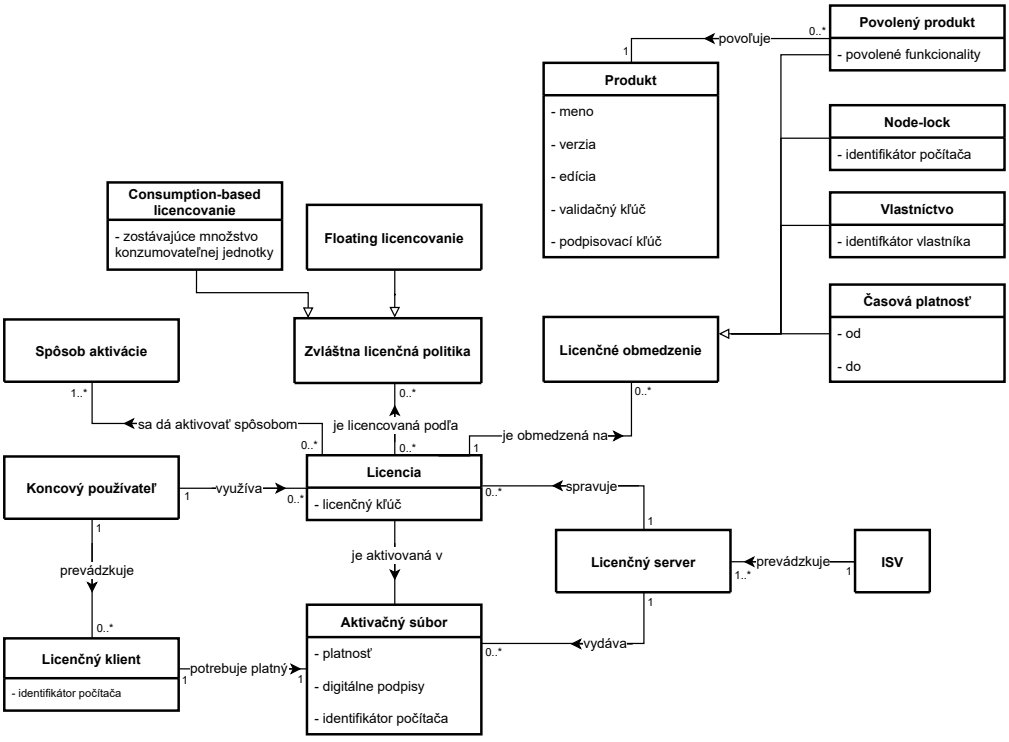
Produkt je defininícia softvérového produktu, ktorý podlieha licencovaniu a môže byť na neho vydaná licencia. *Produktov* s rovnakým menom môže byť viacero, avšak s odlišnou verziou.

Aktivačný súbor predstavuje jednu aktiváciu *licencie*, ktorých môže byť v histórii viacero. Každý *aktivačný súbor* môže byť na strane licenčného server zneplatnení (deaktivácia). Dôležité je, že aktivačný súbor nesie so sebou informácie o licencii, má platnosť a je digitálne podpísaný viacerými podpisovacími kľúčmi (toľko podpisov, koľko má licencia obmedzení typu *povolený produkt*. Platný *aktivačný súbor* s pravým digitálnym podpisom produktu je postačujúca podmienka pre licenčný klient, aby softvér mohol byť považovaný za právoplatne používaný.

V doméne sa nachádzajú niektoré nasledujúce dôležité integritné obmedzenia: *licencia* môže byť aktivovaná iba jedným licenčným klientom – to znamená, že *licencia* môže byť aktivovaná nanajvýš jedným *aktivačný súbor*, ktorého platnosť ešte neskončila. Výnimku tvoria tie aktivačné súbory, ktoré boli označené ako neplatné – *licencia* bola deaktivovaná.

Doménový model podporuje online aktiváciu autorizačným kľúčom / licenčným kľúčom a aj offline aktiváciu. Model zároveň podporuje rôzne nasledovné licenčné modely:

- **Večné licencie** – *licencia* je aktivovaná práve jedným aktivačný súborom pre jeden počítač naveky.
- **Dočasné licencie** – *licencia* môže byť aktivovaná viacerými aktivačnými súbormi na jeden počítač, ktoré ale majú disjunkčnú platnosť (napríklad jeden rok, alebo štrnásť dní, v prípade trial licencie). Licencia obsahuje aspoň jedno obmedzenie *časovej platnosti*.
- **Floating licencie** – *licencia* môže byť aktivovaná viacerými aktivačnými súbormi pre viac počítačov, ktoré ale majú disjunkčnú platnosť iba po dobu používania softvéru. Licencia má politiku *floating licencovania*.
- **Consumption-based licencie** – *licencia* na licenčnom serveri drží v sebe informáciu o kvantite konzumovateľnej jednotky (prostredníctvom politiky *Consumption-based licencovanie* a je aktivovaná viacerými aktivačnými súbormi, ktoré ale majú disjunkčnú a typicky veľmi krátku platnosť. Pri každej aktivácii sa odkrojuje z kvantity konzumovateľnej jednotky licencie. Po vyčerpaní kvantity, *licenciu* nie je možné už viac aktivovať.



Obr. 3.4: Diagram doménového modelu licenčného servera

Kapitola 4

Návrh

4.1 Návrh architektúry

4.1.1 Trojvrstvová architektúra

Trojvrstvová architektúra (Three-tier architecture) je návrhový vzor softvéru pre aplikácie typu klient-server, ktorej hlavným cieľom je rozdeliť aplikáciu na tri modulárne úrovne – používateľské rozhranie, aplikačná logika a dátové uložisko. Výhodou tohto prístupu je, že táto architektúra umožňuje kedykoľvek niektorú z troch vrstiev zameniť, napríklad pokiaľ je nutné zmeniť požiadavky alebo technológie. Architektúra definuje nasledujúce tri vrstvy aplikácie:

- *Prezentačná vrstva* – implementuje používateľské rozhranie, spracováva vstup od používateľa, volá funkcie z logickej vrstvy a prekladá ich výstupy do zrozumiteľnej podoby.
- *Logická vrstva* – spracováva príkazy a vykonáva logické rozhodnutia v závislosti od implementovanej biznis logiky. Táto vrstva taktiež koordinuje dáta medzi prezenčnou a dátovou vrstvou.
- *Dátová vrstva* – získava a ukladá informácie komunikáciou s dátovým úložiskom.

4.1.2 Architektúra licenčného servera

Architektúra univerzálneho licenčného server je implementáciou *trojvrstvovej architektúry* prispôsobená požiadavkam servera. Licenčný server bude implementovať nasledujúce balíčky:

- *Dtos (Data transfer objects)* – balíček obsahuje definície objektov, ktoré sa používajú len na prenos dát medzi klientom a serverov. Tieto objekty sa delia na *vstupné* a *výstupné*. *Vstupné* objekty sú určené na pre-

nos dát smerom od klienta na server a *výstupné* objekty sú určené na opačný smer – prenos dát smerom od servera ku klientovi. Tieto objekty sú súčasťou rozhrania, ktoré klient musí implementovať.

- *Controllers* – balíček obsahujúci vstupné body servera (endpoints) pre spracovania požiadavok od klienta. Táto vrstva sa stará o deserializáciu vstupu od klienta, volaním funkcii logickej vrstvy a serializáciu výstupov funkcií pre klienta.
- *Services* – balíček obsahujúci hlavnú aplikačnú logiku servera – napr. čítanie voľných licencií, vytváranie licencií alebo vydávanie aktivačných súborov.
- *Models* – balíček obsahujúci zdroje a definície ich atribútov – napr. definícia licencie alebo produktu.
- *Repositores* – balíček, ktorý poskytuje rozhranie prístupu k dátovému úložisku servera, nezávisle od technológie úložného priestoru (či už je to databáza alebo súbory).

Poloha jednotlivých balíčkov vrámci vrstiev a smer toku volaní je znázornení na obrázku 4.1.

4.2 REST API špecifikácia

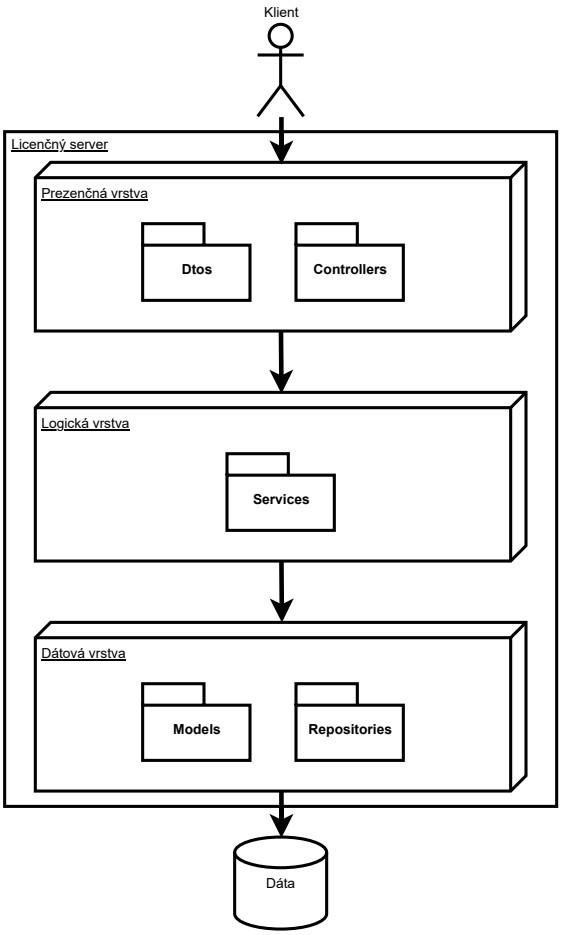
Aplikačné servery sú služby ktoré napĺňajú potreby iným službám alebo aplikáciam. Klientske aplikácie, ktoré využívajú služby aplikačných serverov využívajú na komunikáciu so serverom aplikačné programové rozhranie serveru (API), ktoré ich poskytuje. API sprístupňuje funkcie na interakciu medzi softvérmi a umožňuje im výmenu informácii. Webové API je hlavnou činnosťou podmnožiny serverov – webových serverov. Ich úlohou je počúvať a spracovávať požiadavky klientov prostredníctov webových protokolov.

Architektonický štýl *REST* sa bežne používa pri navrhovaní API pre moderné webové služby. Webové API zodpovedajúce architektonickému štýlu REST sa nazýva *REST API* a je typicky implementované protokolom HTTPS.

REST API pozostáva zo zostavy navzájom prepojených zdrojov (interlinked resources). Táto zostava zdrojov je tiež známa ako model prostriedkov. Zdrojom môžu byť dáta alebo stav aplikácie.

Všetky zdroje majú svoj vlastní identifikátor *URI* a *REST* definuje niekoľko základných metód pre prístup k nim – *GET* (čítanie zdroju), *PUT* (vytváranie zdroju), *POST* (upravovanie zdroju) a *DELETE* (mazanie zdroju). Výhodami konceptu REST je jednoduché a rozšíriteľné rozhranie, jednoduchá implementácia zo strany klienta a transparentnosť.

REST API je typicky implemenované HTTPS protokolom a formát na výmenu dát sa bežne používa JSON. [18]



Obr. 4.1: Diagram implementácie trojvrstvovej architektúry licenčného klienta

4.2.1 Prehľad koncových bodov

GET `/api/Licences`

Parametre: žiadne
Odpoveď: kód: 200 formát: `text/json` schéma: `LicenseDto[]`

GET `/api/Licences/{id}`

Parametre: meno: `id` schéma: `uuid`
Odpoveď: kód: 200 formát: `text/json` schéma: `LicenseDto`

GET `/api/Licences/free`

Parametre: žiadne
Odpoveď: kód: 200 formát: `text/json` schéma: `LicenseDto[]`

GET `/api/Licences/with-owner-id/{ownerId}`

Parametre: meno: `ownerId` schéma: `string`
Odpoveď: kód: 200 formát: `text/json` schéma: `LicenseDto[]`

PUT `/api/Licences`

Parametre: meno: `content` schéma: `CreateLicenseDto`
Odpoveď: kód: 200 formát: `text/json` schéma: žiadna

POST `/api/Licences/activate-license-key/{licenseKey}`

Parametre: meno: `licenseKey` schéma: `string`
meno: `content` schéma: `ActivationRequestFileDto`
Odpoveď: kód: 200 formát: `text/json` schéma: `SignedActivationFileDto`

POST `/api/Licences/activate-id/{id}`

Parametre: meno: `id` schéma: `string`
meno: `content` schéma: `ActivationRequestFileDto`
Odpoveď: kód: 200 formát: `text/json` schéma: `SignedActivationFileDto`

POST `/api/Licences/deactivate-license-key/{licenseKey}`

Parametre: meno: `licenseKey` schéma: `string`
Odpoveď: kód: 200 formát: `text/json` schéma: žadna

GET `/api/Products`

Parametre: žiadne
Odpoveď: kód: 200 formát: `text/json` schéma: `ProductDto[]`

GET `/api/Products/{id}`

Parametre: meno: `id` schéma: `uuid`
Odpoveď: kód: 200 formát: `text/json` schéma: `ProductDto`

PUT `/api/Product`

Parametre: meno: `content` schéma: `CreateProductDto`
Odpoveď: kód: 200 formát: `text/json` schéma: žiadna

4.3 Použité technológie

V nasledujúcej časti sú popisané technológie, ktoré budú použité na základe zadania a na základe dobrej podpory navrhovanej architektúry.

4.3.1 ASP.NET Core

ASP.NET Core je cross-platformový open-source framework pre vytváranie moderných aplikácií v programovacom jazyku *C#*. Framework zahŕňa aj zabudovaný webový server *Kestrel*, unifikovaný spôsob vytvárania webového

API pomocou MVC architektúry a grafického UI pomocou technológie Razor. Okrem iného má zabudovaný dependency injection a rôzne nástroje, ktoré uľahčujú vývoj webovej aplikácie. [19]

4.3.2 Kestrel

Kestrel je zabudovaný webový server v ASP.NET Core frameworku s podporou HTTPS. Keďže webové aplikácie vyčkávajú a reagujú na požiadavky klientov, tak server pracuje na princípe event-driven programovaní – obsahuje hlavnú smyčku v ktorej spracováva udalosti. Pokiaľ nastane nejaká požiadavka zo strany klienta, *Kestrel* zavolá callback funkciu. Ako každý štandardný server, aj *Kestrel* je konfigurovateľný súborom *appsettings.json* – je možné nastaviť hostname, port ale aj user-code parametre. [20]

4.3.3 Microsoft SQL Server

Microsof SQL Server je relačný databázový systém, ktorého primárnou úlohou je ukladať a čítať dáta iných aplikácií v tabuľkách. Manipulácia s dátami prebieha cez dotazovací jazyk *SQL*. Všeobecne, úlohou databázy je dáta poskytovať aj aplikáciam mimo toho istého počítača (cez sieť) ale aj zabraňovať inkonzistencii dát v prípade výpadku elektriny. Výhodou *Microsoft SQL Servera* je vysoká bezpečnosť, vysoký výkon a podpora v *Entity Framework*. Server je určený pre platformy *Windows* a *Linux*. [21]

4.3.4 Entity Framework

Entity Framework Core je ľahká, open-source knižnica pre ASP.NET Core slúžiaca na manipuláciu s dátami v databáze. Knižnica umožňuje jednoduché pripojenie k databázovému systému a mapovanie riadkov tabuliek (ORM) na objekty v C#.

V *Entity Frameworku* sa pristupuje k dátam cez modely a kontexty. Model je trieda v jazyku C# ktorá predstavuje entitu (riadok) v tabuľke databázy. Kontext je objekt, ktorý predstavuje pripojenie k databáze a umožňuje vykonávať manipuláciu s dátami – vytvárať, meniť alebo mazať entity modelu.

Migrácia je kód, ktorý sychronizuje schéma databázy (tabuľky a stĺpce) s modelmi definovanými v kóde aplikácie. [22]

Typický pracovný postup je s knižnicou je:

1. Definícia modelu naprogramovaním triedy s parametrami (členskými premennými).
2. Vytvorenie migrácie cez príkazový riadok a následné pripojenie k databáze a vykonanie migrácie nad ňou.
3. Dotazovanie cez kontext počas behu aplikácie (napríklad vytvorenie novej entity modelu, teda vytvorenie nového riadku v tabuľke).

4.3.5 PKCS #7

PKCS #7 (CMS) je štandard pre digitálne podpisovanie dát definovaný v RFC 5652, ktorý sa používa na digitálne podpisovanie, overovanie podpisu alebo zašifrovanie arbitrárneho obsahu.

Zjednodušená štruktúra syntaxe súboru *PKCS #7* [23] je uvedená na 4.1.

```
  ContentInfoType - identifikátor typu obsahu
  EncapsulatedContent - bajty obsahu
SignerInfos - časť digitálneho podpisu
  DigestAlgorithm - hešovací algoritmus (napr. SHA-1)
  SignatureAlgorithm - algoritmus podpisu (napr. RSA, DSA, ECDSA)
  SignatureValue - bajty digitálneho podpisu
```

Kód 4.1: Registrácie implementácii rozhrania

Kapitola 5

Implementácia

5.1 Diagram tried

Vzhľadom na to, že licenčný server je implementovaný v objektovo orientovanom jazyku, je možné zachytiť štruktúru objektov a vzťahy medzi nimi takzvaným diagramom tried – typ diagramu v jazyku *UML*.

Diagram tried je rozdelený na tri menšie diagramy. Každý z diagramov prislúcha jednej z vrstiev architektúry licenčného servera. Diagram na obrázku A.1 reprezentuje *prezentačnú vrstvu*, diagram na obrázku B.1 reprezentuje *logickú vrstvu* a diagram na obrázku C.1 znázorňuje *dátovú vrstvu* licenčného servera.

Pri návrhu tried aplikácie sa vychádzalo primárne z doménového modelu a z navrhovanej architektúry licenčného servera.

5.2 Databázová realizácia

Dátová vrstva definuje jednotlivé zdroje dátovými modelmi – štruktúry, ktoré reprezentujú entity v doméne, ktoré predmetom uchovávania v dátovom úložisku, napr. `License`, `Product`, `Constraint` a iné.

Na prístup k týmto zdrojom z *logickej vrstvy* sa používajú rozhrania `ILicenseRepository`, `IProductRepository`,... Tieto objekty samé o sebe implementáciu nemajú. Implementácia repozitárov je dodaná triedami `LicenseProductDbRepository`, `ProductDbRepository`,..., ktoré implementujú čítanie/zápis entít do databázy prostredníctvom triedy `UlsDbContext`, ktorá drží pripojenie k databáze a vykonáva *SQL dotazy*.

Štruktúra databázy je zachytená diagramom na obrázku D.1.

5.2.1 Dedičnosť v databáze

V doménovom modeli a v návrhu modelov entít sa nachádza dedičnosť: `Constraint` a `Policy` sú abstraktné triedy, ktoré môžu mať viac derivo-

vaných tried. Keďže jazyk *SQL* priamo dedičnosť nepodporuje, bolo je implementovať návrhový vzor *Table-per-hierarchy*. Tento návrhový vzor vkladná všetky podtypy do jednej spoločnej tabuľky a definuje k tomu špeciálny stĺpec, ktorý slúži na rozoznanie typu, ktorý sa v riadku nachádza. Tento návrhový vzor bol zvolený pre jeho vyšší výkon a pre vyhýbanie sa duplikovania dát oproti iným návrhovým vzorom (napr. *Table-per-type*, *Entity type hierarchy mapping*,...). [24]

5.2.2 Migrácie

Pri spustení servera *Entity Framework* vykoná *migrácie* – pokiaľ je databáza prázdna, tak vytvorí tabuľky v databáze na základe atribútov entitných modelov. Pokiaľ databáza nie je prazdná, vykoná zmeny v schéme databázy, čím odstráni nekonzistentnosť databázy s aplikáciou. História úprav databázovej schémy migráciami sa zapisuje do tabuľky `__EFMigrationsHistory`. [25]

5.3 Dependency injection

Licenčný server podporuje *dependency injection* pomocou frameworku *ASP.NET Core*. Úlohou tejto technológie je dodať triedam nakonfigurované implementácie ich závislostí. Táto technológia umožňuje veľmi jednoducho nahradiť komponenty licenčného servera. Pri štarte servera je vykonaná registrácia implementácie rozhrania tak, ako je uvedené v 5.1.

```
builder.Services.AddScoped<IProductRepository, ProductDbRepository>();
builder.Services.AddScoped<ILicenseRepository, LicenseDbRepository>();
...
builder.Services.AddScoped<ILicenseService, DefaultLicenseService>();
builder.Services.AddScoped<IProductService, DefaultProductService>();
builder.Services.AddScoped<IPolicyService, DefaultPolicyService>();
...
```

Kód 5.1: Registrácie implementácii rozhrania

Vo všeobecnosti platí, že všade sa používajú rozhrania a tým pádom vymeniť chovanie licenčného servera alebo nahradiť technológiu úložného priestora je veľmi jednoduché.

5.4 Autorizácia

Autentifikácia tokenu je proces verifikácie platnosti tokenu klienta. Klient získa prístupový token z externého autorizačného servera.

Každá požiadavka, ktorá príde na licenčný server, je skontrolovaná. Ak sa nájde v požiadavke platný token, požiadavka sa spracuje. Ak sa nenájde

žiadny token alebo je token neplatný, požiadavka sa zamietne s odpoveďou 401 Unauthorized.

Licenčný server podporuje autentifikáciu tokenu cez protokol *OpenID Connect*.

Server pri štarte sa pokúsi získať metadáta z nakonfigurovaného autorizačného servera (nazývaného aj autorita). Tieto metadáta obsahujú verejné kľúče a ďalšie podrobnosti potrebné na overenie tokenov. [26]

5.4.1 Prístupový token

Prístupový token klienta autorizovaného protokolom *OpenID Connect* je vo formáte *JWT*. Samotný token sa skladá z troch častí [27]:

- *Header* – obsahuje informáciu o tom, akým hešovacím algoritmom bol spočítaný digitálny podpis prístupového tokenu.
- *Payload* – obsahuje informácie o čase vydania tokenu, identifikátor subjektu (položka *sub*) a zoznam jeho rolí (položka *roles*).
- *Signature* – obsahuje digitálny podpis kombinácie častí *header* a *payload*.

Licenčný server prístupový token rozbalí a sprístupní ho na čítanie vstupným bodom serveru.

5.4.2 Zabezpečenie vstupných bodov

Každý vstupný bod (metóda, ktorá je zavolaná po prijatí požiadavky) je ochránený podmienkou platného prístupového tokenu pomocou anotácie `Authorize`. Anotácia má nepovinný atribút potrebných rolí `Roles`. Kontrola platnosti tokenu prebieha automaticky pred zavolaním koncového bodu – overuje ho *ASP.Net Core* framework.

Koncový používateľ nemusí mať špecifikovanú žiadnu rolu, ale musí mať platný prístupový token.

Administrátor licencií ISV musí mať platný prístupový token, ktorý obsahuje rolu `ULSAdmin`.

Príklad zabezpečneného vstupného bodu, ktorý je povolený iba pre administrátorov licencií ISV je uvedený v 5.2.

5.5 Rozšíriteľnosť licencií

Vlastnosti licencie sú odzrkadlené členskými premennými triedy `License`. Okrem identifikátora a licenčného kľúča (sekundarný unikátny identifikátor) každá licencia disponuje množinou obmedzení – `Contraint` a množinou zvláštnych politík – `Policy`. Rozšírenie licencie o ďalšie informácie je možné:

```
[HttpPut]
[Authorize(Roles = "ULSAdmin")]
public IActionResult CreateLicense([FromBody] CreateLicenseDto dto)
{
    License license = _licenseService.CreateLicense(dto);
    return CreatedAtAction(
        nameof(GetLicense),
        new { id = license.Id }, license.ToDto()
    );
}
```

Kód 5.2: Príklad zabezpečeného vstupného bodu

- derivovaním abstraktnej triedy `Constraint` – pokiaľ sú pridané informácie statické a obmedzujú použitie licencie aj na strane licenčného klienta,
- derivovaním abstraktnej triedy `Policy` – pokiaľ pridané informácie slúžia iba serverovej časti alebo
- pridaním novej členskej premennej medzi existujúce.

Príklad možného rozšírenia by mohlo byť pridanie *application-specific dát* do licencie. Keďže sa jedná o dáta, ktoré slúžia aplikácii a teda sú čítané licenčným klientom, rozšírenie by bolo možné implementovať zavedením triedy `ApplicationDataConstraint` derivovaním triedy `Constraint` tak, ako je vedené v 5.3.

```
// existujúca trieda
public abstract class Constraint
{
    public Guid Id { get; set; }
    public virtual License License { get; set; }
}

// rozšírenie licencie o aplikačné dáta
public class ApplicationDataConstraint : Constraint
{
    public byte[] Data { get; set; }
}
```

Kód 5.3: Príklad licenčného obmedzenia

5.6 Vydávanie aktivačných súborov

Aby pre licenciu mohol byť vydaný aktivačný súbor s digitálnym podpisom, licencia musí mať priradený `ProductConstraint`. O toto integritné obmedzenie sa stará aplikačná logika, ktorá neumožňuje vytvoriť licenciu bez priradeného produktového obmedzenia. Model obmedzení licencie avšak dovoľuje mať licencii aj viacero priradených `ProductConstraint` – v tom prípade sa jedná o takzvanú *multilicenciu* – licencia, ktorá je použiteľná pre viacero softvérových produktov. Avšak každý `Product` má svoj vlastný podpisovací kľúč a algoritmus.

Aby bol prípad *multilicencií* licenčným serverom pokrytý, štruktúra aktivačného súboru je koncipovná tak, ako je uvedené na 5.4:

```
public class ActivationFileDto
{
    public Guid Id { get; set; }
    public DateTime ValidFrom { get; set; }
    public DateTime ValidTo { get; set; }
    public LicenseDto License { get; set; }
}

public class SignedActivationFileDto
{
    public byte[] ActivationFile { get; set; }
    public ICollection<byte[]> ActivationFileSignatures { get; set; }
}
```

Kód 5.4: Štruktúra aktivačného súboru

Aktivačný súbor, ktorý licenčný server pre klienta vydá je zaserializovaný objekt typu `SignedActivationFileDto` vo formáte *JSON*. Tento objekt obsahuje reťazec bajtov v ktorom je zaserializovaný objekt typu `ActivationFileDto` vo formáte *JSON*. Dôvod, prečo je táto položka reťazec bajtov a nie priamo objekt typu `ActivationFileDto` (napríklad tak ako jeho vnútorná položka `License`) je, že vypočítať heš prameho typu nie je možné. Heš je nutné počítať pre vyrátanie digitálnych podpisov. Digitálnych podpisov (položka `ActivationFileSignatures`) je práve toľko, na koľko produktov sa licencia viaže – každý podpis je vytvorený iným podpisovacím kľúčom. Pokiaľ licenčný klient nájde medzi tymito podpismi aspoň jeden taký, ktorému dôveruje (produkt má v sebe integrovaný validačný kľúč rovnaký ako na serveri) tak licenciu považuje za platnú.

Pre samotné vyrátavanie digitálnych podpisov sú použité kryptografické algoritmy zo `System.Security.Cryptography`, tak ako na 5.5.

Poznámka: licenčný server podporuje RSA a ECDsa pre asymetrické algoritmy. Kód uvedený v 5.5 je zjednodušený a jeho skutočná implementácia

```
AsymmetricAlgorithm aa =
    AsymmetricAlgorithm.Create(product.AsymmetricAlgorithm);
aa.ImportPkcs8PrivateKey(product.SigningKey);

byte[] activationFileSignatureForProduct = aa.SignData(
    activationFileBytes,
    new HashAlgorithmName(product.HashAlgorithm)
);

activationFileSignatures.Add(activationFileSignatureForProduct);
```

Kód 5.5: Digitálne podpisovanie aktivačného súboru

sa nachádza v triede `DefaultActivationFileIssuingService`.

5.7 Konfigurácia a nasadenie

Konfigurácia licenčného servera sa vykonáva rovnako ako každá iná ASP.NET Core aplikácia pomocou jedného alebo viacerých poskytovateľov konfigurácie. Licenčný server číta konfiguračné premenné z párov kľúč-hodnota pomocou rôznych zdrojov konfigurácie, ako napríklad [28]:

- súbory nastavení `appsettings.json`,
- enviromentálne premenné,
- argumenty príkazového riadku a pod.

5.7.1 Databáza

Databázu SQL je možné nakonfigurovať pomocou *Connection Stringu* – reťazec, ktorý obsahuje IP adresu databázy, meno/heslo a názov databázy. Tento reťazec je poskytovaní každým databázovým serverom a jeho formát je závislý od implementácie výrobcom. Licenčný server prijíma reťazce pripojenia *Microsoft SQL Serveru*.

```
"ConnectionString": "<retˈazec_pripojenia>"
```

Kód 5.6: Konfigurácia databázového pripojenia

5.7.2 Autorizačný server

Napojenie licenčného servera na externý autorizačný server podporujúci *OpenID Connect* sa vykonáva poskytnutím jeho *URL* a *Audiencie*.

```
"AuthorizationServer": "https://<autorizačný_server>/oauth2/default",
"AuthorizationServerAudience": "<audiencia>"
```

Kód 5.7: Konfigurácia autorizačného servera

5.7.3 Platnosť aktivačného súboru

Každý aktivačný súbor je vydaný iba na obmedzenú dobu. Dĺžka tejto platnosti závisí od licenčného modelu a dá sa nakonfigurovať
vo formáte `<dni>:<minúty>:<sekundy>`.

```
"ActivationFileValidityLimitedValidityLicense": "14:00:00",
"ActivationFileValidityFloatingLicense": "00:05:00",
"ActivationFileValidityConsumptionBasedLicensing": "00:01:00"
```

Kód 5.8: Konfigurácia doby platnosti digitálnych podpisov

5.8 Nasadenie

Licenčný server je nutné zostaviť v *Release* konfigurácii pre cieľovú platformu (napr. *Windows Server - x64* alebo *Linux x86*) vývojový prostredím *Visual Studio* alebo príkazom `dotnet` v termináli. Výsledné spustiteľné binárne súbory sú sebestačné a obsahujú zabudovaný webový server licenčného servera.

5.8.1 Docker

Alternatívne, licenčný server podporuje nasadenie cez *Docker container*. V adresári so zdrojovými súbormi sa nachádza súbor `Dockerfile` s obsahom uvedeným v 5.9.

Pre spustenie *Docker containera* je nutné sa v termináli nachádzať v adresári s *Dockerfile* a spustiť príkazy uvedené v 5.10:

```
FROM mcr.microsoft.com/dotnet/aspnet:6.0 AS base
WORKDIR /app
EXPOSE 80
EXPOSE 443

FROM mcr.microsoft.com/dotnet/sdk:6.0 AS build
WORKDIR /src
COPY ["UniversalLicenseServer.csproj", "."]
RUN dotnet restore "./UniversalLicenseServer.csproj"
COPY . .
WORKDIR "/src/."
RUN dotnet build "UniversalLicenseServer.csproj" -c Release -o /app/build

FROM build AS publish
RUN dotnet publish "UniversalLicenseServer.csproj" -c Release -o /app/publish

FROM base AS final
WORKDIR /app
COPY --from=publish /app/publish .
ENTRYPOINT ["dotnet", "UniversalLicenseServer.dll"]
```

Kód 5.9: Konfiguračný súbor Dockerfile

```
$ docker build -t universal-license-server .
$ docker run -d -p 8080:80 --name uls universal-license-server
```

Kód 5.10: Príkazy pre spustenie servera v Docker

Kapitola 6

Zhodnotenie

6.1 Pohľad na prototyp z bezpečnostného hladiska

V tejto časti práce je diskutovaná bezpečnosť implementovaného prototypu licenčného servera. Existuje niekoľko scenárov, ktoré by mohli viesť k neoprávnenému používaniu licencovaného softvéru aj napriek integrácii prototypu licenčného servera uvedeného v tejto práci.

6.1.1 Prelomenie digitálneho podpisu

Prelomením digitálneho podpisu sa myslí vypočítanie súkromného kľúču asymetrického šifrovacieho algoritmu použitého na vytvorenie digitálneho podpisu aktivačného súboru.

Navrhovaný licenčný server podporuje algoritmus *RSA*, ktorého útok je založený na probléme faktorizácie celočísla. Je známe, že momentálne neexistuje algoritmus, ktorý by vedel vyriešiť problém faktorizácie celočísla v polynomiálnom čase. Najoptimálnejší dostupný algoritmus, *sito všeobecného číselného poľa (GNSF)*, má subexponenciálnu zložiťosť. Pri dostatočne dlhom súkromnom kľúči, napríklad 2048 bitov a viac, je výpočetne nemožné útok prelomenia digitálne podpisu previesť. [29]

Licenčný server podporuje aj algoritmus *ECDsa*, ktorého útok je založený na probléme diskrétneho logaritmu. Podobne ako u *RSA*, prelomenie súkromného kľúču *ECDsa* je výpočtovo nemožné. [29]

6.1.2 Únik súkromného kľúču autorizačného servera

Únik súkromného kľúču autorizačného servera znamená, že útočník získal kľúč, pomocou ktorého si dokáže vytvoriť falošný autorizačný token. Pokiaľ by si vytvoril token, ktorého *subjekt* je niekto s administrátorskými rolami, vie použiť všetky zabezpečené volania dostupné iba pre administrátora ISV.

Útočník by si dokázal vytvoriť licenciu a vydať si na ňu aktivačný súbor a tým používať licencovaný softvér neoprávnene.

Avšak väčšinou tento útok väčšinou nespôsobuje dlhotrvajúce škody. Databáza licenčného servera je obnoviteľná zo záloh a autorizačný server môže pomerne rýchlo súkromný kľúč zmeniť.

Dôležitým zabezpečovacím prvkom navrhovaného licenčného servera je, že aj keď má osoba administrátorské oprávnenia, nedokáže prečítať súkromný kľúč produktu z databázy – neexistuje preň koncový bod, ktorý by vrátil jeho hodnotu.

6.1.3 Únik súkromného kľúču produktu

V prípade, že by útočník získal prístup k čítaniu databázy navrhovaného licenčného servera, prečítaním stĺpcov `SigningKey`, `AsymmetricAlgorithm`, `HashAlgorithm` a znalosťou štruktúry aktivačného súboru, získava všetky informácie k tomu, aby si vyrobil ľubovolný platný aktivačný súbor, ktorý by bol prijatý originálnym licencovaným softvérom od ISV.

Útok tohto typu predstavuje najhorší možný scenár, pretože všetky predané licencie na daný produkt prestávajú nadobúdať zmysel – licencovaný softvér môže neoprávnene používať každý, bez ohľadu na to, či má alebo nemá zakúpenú licenciu.

V prototype je použité ukladanie súkromného kľúča produktu do databázy vo forme *plaintext*, čo uľahčuje útok tohto typu. V komerčnom použití sa väčšinou tieto kľúče neukladajú do databázy, ale využívajú sa takzvané *bezpečné úložiská kľúčov*, ktoré sú oddelené od licenčného servera a ponúkajú mu službu šifrovania, ako napríklad *Azure Key Vault*. [30]

6.1.4 Cracknutie produktu

Navrhovaná architektúra licenčného systému funguje na tej báze, že produkt v sebe obsahuje licenčný klient a verejný kľúč produktu. Tento verejný kľúč použije licenčný server na verifikáciu pravosti aktivačného súboru – teda či aktivačný súbor naozaj pochádza z dôveryhodného zdroja – licenčného servera.

Pod pojmom *cracknutie produktu* sa rozumie nahradenie verejného kľúča za iný alebo obídenie licenčnej ochrany.

Zabezpečiť odolnosť voči tomuto typu útoku, narozdiel od ostatných, nie je možné. V momente keď sú binárne súbory produktu vydané na verejnosť, ISV nemá pod kontrolou ich neautorizované modifikovanie. Avšak ISV môže rôznymi metódami, ako napríklad obfuskáciou kódu, odtialiť vydanie upravených binárnych súborov produktu tretími stranami.

Jediným možným spôsobom, ako zabrániť tomuto typu útoku je neposkytovať binárne súbory koncovým používateľom a aplikáciu ponúkať ako vzdialenú službu – kritické súčasti aplikácie sú vykonávané servermi ISV.

6.2 Pohľad na prototyp z prispôsobiteľnosti

Prispôsobiteľnosť licenčného serveru znamená, že si ISV dokáže jednoducho upraviť licenčný server tak, aby bol v súlade s jeho licenčnou politikou alebo aby poskytoval takú funkcionalitu, ktorú momentálne neposkytuje.

6.2.1 Podpora licenčnej politiky

Server disponuje podporou niekoľkých existujúcich licenčných politík, ktoré sú použiteľné ihneď. Pokiaľ ISV disponuje takou licenčnou politkou, ktorá nie je v licenčnom serveri implementovaná, je nutné pridať nové licenčné obmedzenie `Constraint` alebo zvláštnu licenčnú politku `Policy`, vrátane implementácie chovania.

Použitý návrhový vzor *Table-per-hierarchy* má tú výhodu, umožňuje pridávať nové podtriedy `Constraint` a `Policy` bez zásahu do existujúcich dát. Podobne, umožňuje jednoducho odstranovať podtriedy a dáta iba týchto podtried.

6.2.2 Implementácia novej funkcionality

Prototyp licenčného servera využíva pre celú aplikačnú logiku *rozhrania* a *dependency injection*. To umožňuje jednoduché nahradenie aplikačnej logiky. Použitie technológie *ASP.Net Core* zabezpečilo jednoduchú správu koncový bodov (endpoint) a pridanie ďalšieho koncového bodu do podtried typu `ControllerBase` neovplyní zvyšnú funkcionalitu licenčného servera. Pridanie novej entity modela je rovnako používateľsky prívetivé – po zadefinovaní štruktúry modelovej triedy si server sám vyrobí a vykoná *migráciu* – prispôsobí databázu bez straty aktuálnych dát.

6.3 Porovnanie s existujúcimi riešeniami z rešerše

Open-source riešenia *OLM* a *License3j* sú založené na obdobnom princípe vydávania aktivačných súborov ako navrhovaný prototyp licenčného servera – digitálne podpisovanie aktivačných súborov a validácia klientom. Navrhovaný prototyp licenčného servera avšak narozdiel od existujúcich riešení podporuje aj multilicencie – licencia, ktorá udeľuje právo na používanie viacerých produktov pomocou viacerých digitálnych podpisov. Rovnako nutné je poznamenať, že uvedené existujúce open-source riešenia nepredstavujú plnohodnotný licenčný server, iba utilitu/knižnicu pre vydávanie licencií – nekomunikujú priamo s licenčným klientom. Rovnako neumožňujú správu licencií. Avšak ich výhodou oproti prototypu licenčného servera je prítomnosť referenčného licenčného klienta – ten je prítomný aj pri riešeniach *Licence4j* a *FlexNet Publisher*.

Navrhovaný prototyp licenčného servera sa ponúkanou funkcionalitou viac približuje riešeniam *Licence4j* a *FlexNet Publisher* – jedná sa o plnohodnotné licenčné servery so správou licencií. Výhodou *Licence4j* oproti prototypu je prítomnosť grafického rozhrania pre správu licencií. Výhodou *FlexNet Publisher* je priama podpora *dôveryhodného úložiska* pre aktivačný súbor, alebo aj zabezpečenie dostupnosti pomocou *Three-server redundancie*. je nutné spomenúť, že ani jedno z existujúcich riešení nedisponuje licenčnou politikou *consumption-based licencie*, ktorá je ale v prototype podporovaná.

Záver

Hlavným cieľom práce bolo podľa tradičných postupov softvérového inžinierstva analyzovať, navrhnúť vhodnú architektúru licenčného servera a implementovať jeho prototyp. Navrhovaná architektúra dbá ohľad na prispôsobiteľnosť, bezpečnosť a aj univerzálnosť.

Implementovaný prototyp licenčného servera spĺňa všetky funkčné a nefunkčné požiadavky priority *must have*, niekoľko požiadaviek priority *could have* a *should have*. Licenčný server implementuje špecifikované *REST API rozhranie* a je pripravený obsluhovať licenčného klienta. Dôležitou súčasťou prototypu je schopnosť vydávať aktivačné súbory licencií s nasledujúcimi licenčnými politikami: *večné licencie, dočasné licencie, node-bound licencie, floating licencie a consumption-based licencie*.

V tejto bakalárskej práci bol navrhovaný licenčný server zhodnotený z bezpečnostného hľadiska a poskytuje najvyššiu možnú ochranu pred neoprávneným použitím konvenčného softvéru. Zároveň navhnutá architektúra sa javí dostatočne flexibilná a dobre reaguje na možné úpravy licenčných politík. Prototyp navrhnutého licenčného servera sa požiadavkami a funkcionalitou približuje ku komerčným, plnohodnotným licenčným serverom. Okrem toho, licenčný server disponuje prípravou na nasadenie cez *Docker*. Týmto boli zadané požiadavky tejto bakalárskej práce naplnené.

Zdrojové kódy implementovaného prototypu licenčného servera sú dostupné na priloženom CD (príloha F) a v nasledujúcom repozitári:
`https://gitlab.fit.cvut.cz/balazric/universallicenseserver`.

Literatúra

[1] License4j: *License4j License Manager User Guide*. Dostupné z: https://www.license4j.com/documents/LICENSE4J-License-Manager-User-Guide.pdf

[2] Revenera: *FlexNet Publisher 2021 R3 (11.18.2) License Administration Guide*. Dostupné z: https://www.minitab.com/content/dam/www/en/uploadedfiles/documents/license-management/FlexNetLicenseAdminGuide.pdf

[3] Nussbaum, J. L.: Apple Computer, Inc. v. Franklin Computer Corporation Puts the Byte Back into Copyright Protection for Computer Programs. Dostupné z: https://digitalcommons.law.ggu.edu/ggulrev/vol14/iss2/3/

[4] Software Licensing – Why Is It So Important? [Guide for ISVs]. Dostupné z: https://www.10duke.com/software-licensing/

[5] Laurent, A. M. S.: *Understanding Open Source and Free Software Licensing*. O'Reilly Media, Inc., ISBN 9780596005818.

[6] Mirabella, R.: License Management: How Developers Control Software Licensing. 1999. Dostupné z: https://www.basis.com/sites/basis.com/advantage/mag-v3n2/globetrotter.html

[7] Andrews, L.: Subscription VS. Perpetual License: Why subscription software is actually cheaper.

[8] Holý, V.: *License manager – webová aplikace. Bakalářská práce*. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2021.

[9] Gabrielse, Z.: How to Generate Cryptographically Secure License Keys in 2022. 2021. Dostupné z: https://keygen.sh/blog/how-to-generate-license-keys-in-2021/

[10] Messier, J. V. M.: *Secure Programming Cookbook for C and C++*. O'Reilly Media, Inc., ISBN 9780596003944.

[11] Revenera: Macrovision's New FlexNet Publisher Version 11 Platform Licensed by Adobe. Dostupné z: https://www.revenera.com/about-us/press-center/flexnet-publisher-platform-licensed-adobe

[12] SolidWorks: SolidNetWork Licensing. Dostupné z: http://help.solidworks.com/2021/english/SolidWorks/install_guide/c_FLEXnet_publisher_lic_mgt.htm

[13] Hexagon: Hexagon Knowledgebase. Dostupné z: https://support.hexagonmi.com/s/article/How-do-I-update-to-the-latest-floating-license-server-software-1527312551188

[14] Contini, G.: Licensecc: a C++ software license manager. 2020. Dostupné z: https://open-license-manager.github.io/licensecc/

[15] Verhas, P.: License3j Free License management for Java. Dostupné z: https://github.com/verhas/License3j

[16] Agile, A. A.: Prioritization using MoSCoW. Dostupné z: https://comp.anu.edu.au/courses/comp3120/local_docs/readings/Prioritization_using_MoSCoW_AllAboutAgile.pdf

[17] Wikipedia: FURPS. Dostupné z: https://cs.wikipedia.org/wiki/FURPS

[18] Masse, M.: *REST API Design Rulebook*. O'Reilly Media, Inc., ISBN 9781449310509.

[19] Microsoft: Overview to ASP.NET Core. Dostupné z: https://docs.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-6.0

[20] Microsoft: Kestrel web server implementation in ASP.NET Core. Dostupné z: https://docs.microsoft.com/en-us/aspnet/core/fundamentals/servers/kestrel?view=aspnetcore-6.0

[21] Microsoft: SQL Server technical documentation. Dostupné z: https://docs.microsoft.com/en-us/sql/sql-server/?view=sql-server-ver15

[22] Microsoft: Entity Framework documentation. Dostupné z: https://docs.microsoft.com/en-us/ef/

[23] Housley, R.: Cryptographic Message Syntax (CMS). Dostupné z: https://datatracker.ietf.org/doc/html/rfc5652

[24] Microsoft: Table-per-hierarchy and discriminator configuration. Dostupné z: https://docs.microsoft.com/en-us/ef/core/modeling/inheritance

[25] Microsoft: Migrations Overview. Dostupné z: https://docs.microsoft.com/en-us/ef/core/managing-schemas/migrations/?tabs=dotnet-core-cli

[26] Barbettini, N.: Token Authentication in ASP.NET Core 2.0 - A Complete Guide. Dostupné z: https://developer.okta.com/blog/2018/03/23/token-authentication-aspnetcore-complete-guide

[27] Jones, M.: JSON Web Token (JWT). Dostupné z: https://datatracker.ietf.org/doc/html/rfc7519

[28] Microsoft: Configuration in ASP.NET Core. Dostupné z: https://docs.microsoft.com/en-us/aspnet/core/fundamentals/configuration/?view=aspnetcore-6.0

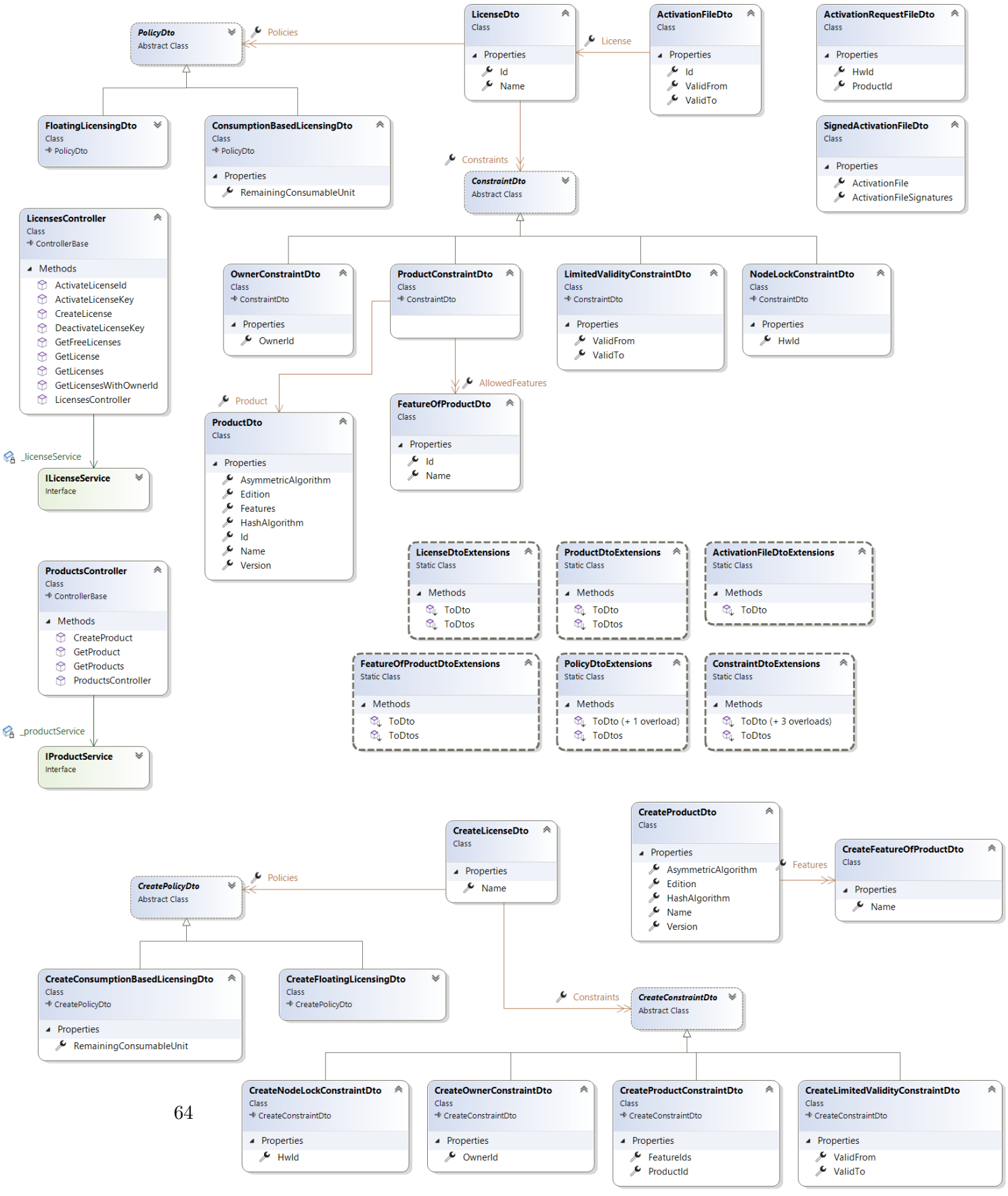
[29] Anish, N.: *The Modern Cryptography Cookbook*. Lightning Source Inc., ISBN 9781718104716.

[30] Microsoft: Azure Key Vault basic concepts. Dostupné z: https://docs.microsoft.com/en-us/azure/key-vault/general/basic-concepts

Dodatok **A**

Diagram tried prezentačnej vrstvy

Diagram tried prezenčnej vrstvy prototypu licenčného servera je zachytený na obrázku D.1. Obrázok je súčasťou priloženého CD vo formáte JPG.

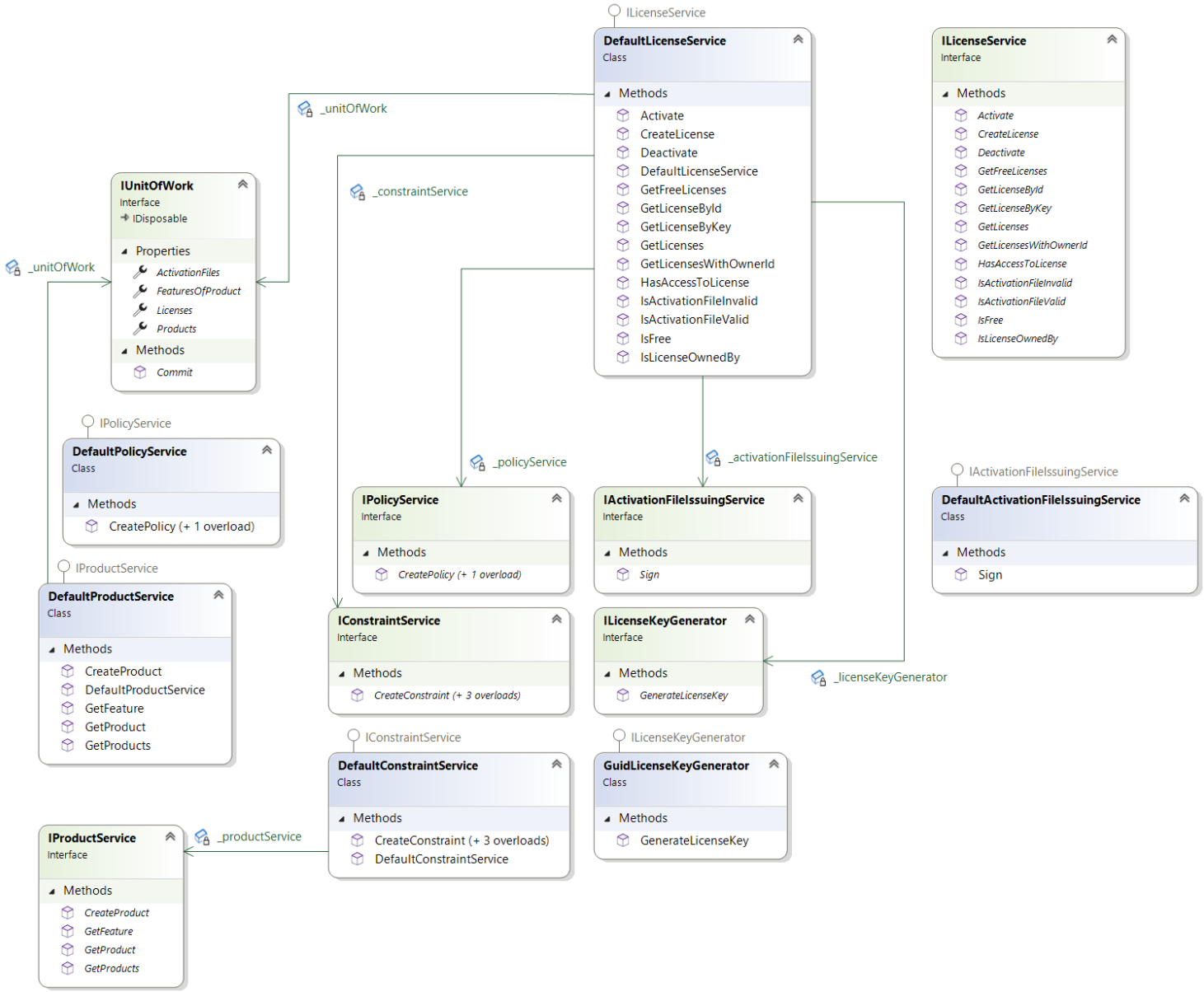


Obr. A.1: Diagram tried prezentačnej vrstvy

Dodatok B

Diagram tried aplikačnej vrstvy

Diagram tried aplikačnej vrstvy prototypu licenčného servera je zachytený na obrázku D.1. Obrázok je súčasťou priloženého CD vo formáte JPG.

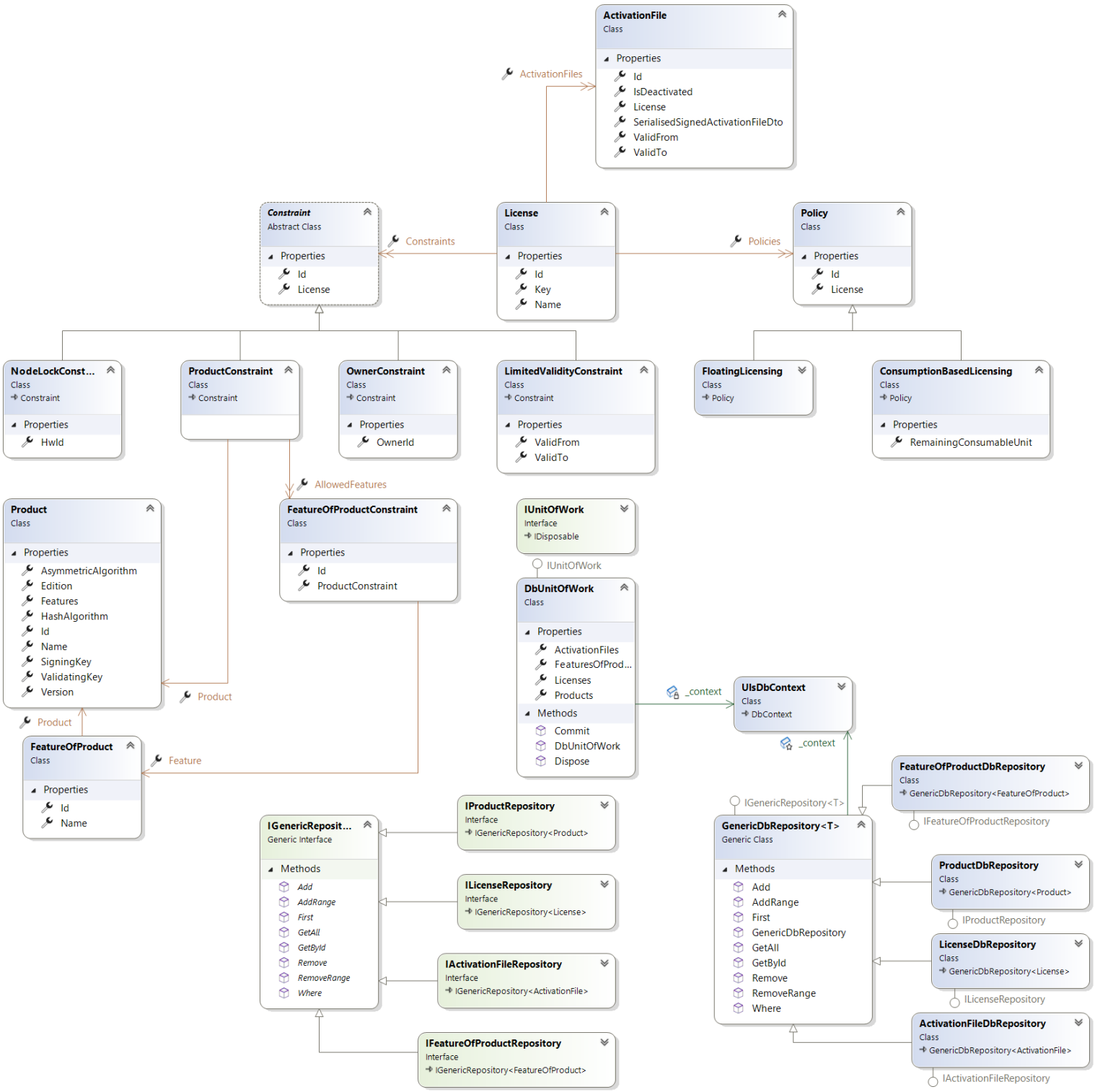


Obr. B.1: Diagram tried aplikačnej vrstvy

Dodatok C

Diagram tried dátovej vrstvy

Diagram tried dátovej vrstvy prototypu licenčného servera je zachytený na obrázku D.1. Obrázok je súčasťou priloženého CD vo formáte JPG.

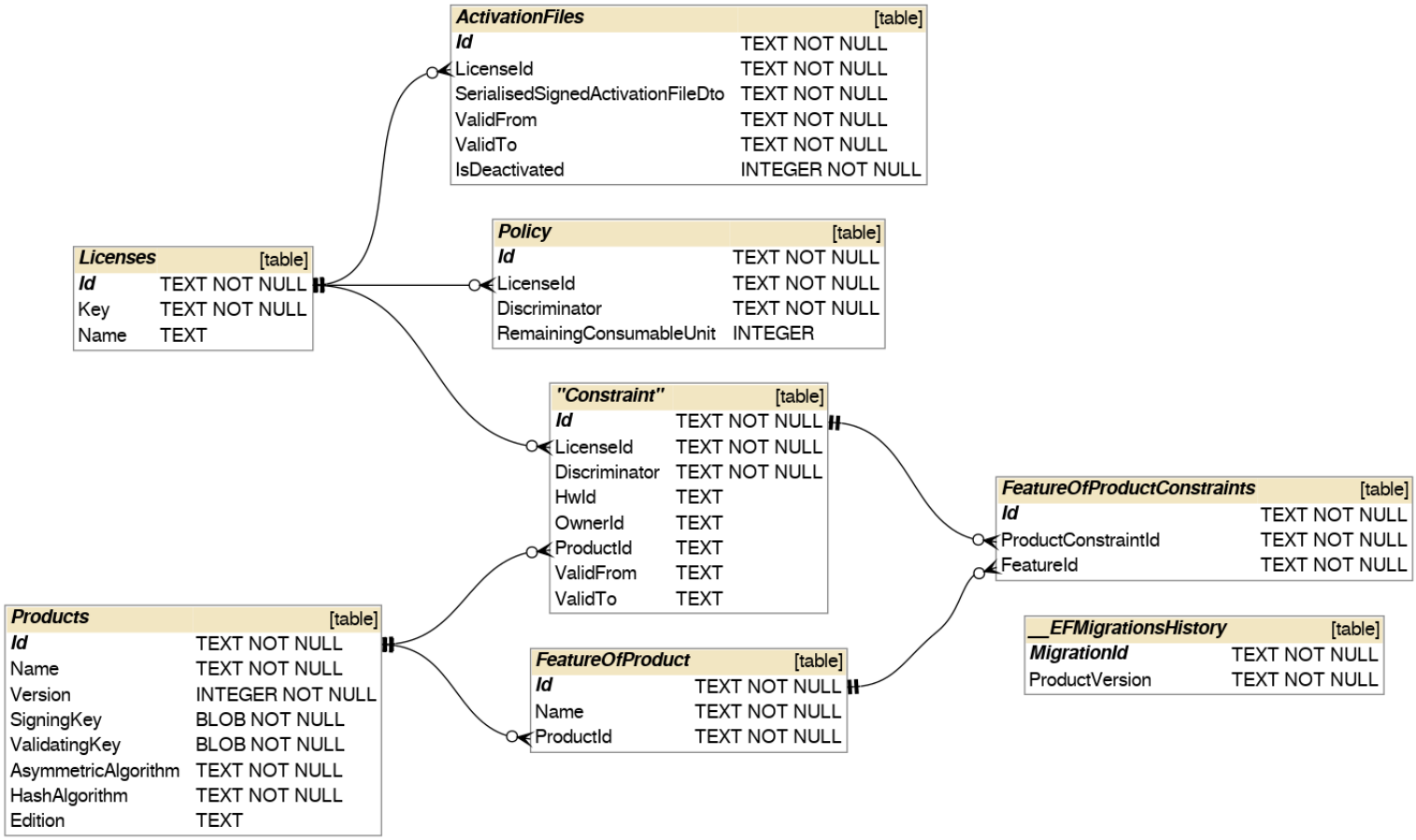


Obr. C.1: Diagram tried dátovej vrstvy

Dodatok D

Databázový diagram

Štruktúra databázy prototypu licenčného servera je zachytená diagramom na obrázku D.1. Obrázok je súčasťou priloženého CD vo formáte JPG.



Obr. D.1: Databázový diagram

Dodatok E

Zoznam použitých skratiek

API Application programming interface

CMS Cryptographic Message Syntax

CPU Central processing unit

CRM Customer relationship management

CRUD Create-read-update-delete

DTO Data Transfer Object

EULA End-user license agreement

FOSS Free and open-source software

FP Funkčné požiadavky

GUI Graphical user interface

HTML HyperText Markup Language

HTTPS Hypertext Transfer Protocol Secure

ID Identifikačné číslo

ISV Independent software vendor

JS JavaScript

LAN Local area network

MTBF Mean time between failures

NP Nefunkčné požiadavky

ORM Object–relational mapping

PKV Partial key verification

PP Prípady použitia

REPL Read–eval–print loop

REST Representational state transfer

SaaS Software as a service

SQL Structured Query Language

TCP/IP Transmission Control Protocol/Internet Protocol

TPM Trusted Platform Module

UML Unified Modeling Language

URL Uniform Resource Locator

USB Universal serial bus

Dodatok F

Obsah priloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `bin` ..... adresár so spustiteľnou formou implementácie
- `src`
  - `impl` ..... zdrojové kódy implementácie
  - `diagrams` ..... návrhové a implementačné diagramy
  - `thesis` ..... zdrojová forma práce vo formáte LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce vo formáte PDF
  - `thesis.ps` ..... text práce vo formáte PS