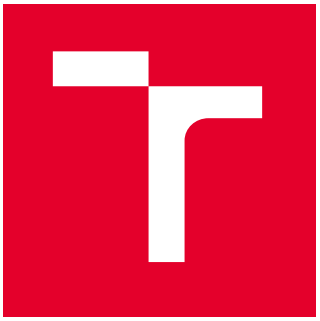


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ

DEPARTMENT OF TELECOMMUNICATIONS

EXPERIMENTÁLNÍ INFRASTRUKTURA PRO ELEKTRONICKÝ GEOKEŠING

EXPERIMENTAL INFRASTRUCTURE FOR ELECTRONIC GEOCACHING

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Pavel Škrhák**
AUTHOR

VEDOUCÍ PRÁCE — **doc. Ing. Karel Burda, CSc.**
SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní program **Informační bezpečnost**

Ústav telekomunikací

Student: Bc. Pavel Škrhák ***ID:*** 211583

Ročník: 2 ***Akademický rok:*** 2022/23

NÁZEV TÉMATU:

Experimentální infrastruktura pro elektronický geokešing

POKYNY PRO VYPRACOVÁNÍ:

V rámci diplomové práce nastudujte a popište infrastrukturu elektronického geokešingu podle [1]. Na tomto základě navrhněte a prakticky zrealizujte infrastrukturu, s jejíž pomocí koncept elektronického geokešingu ověřte. Součástí ověřovací infrastruktury bude navíc i aplikace pro objednávku karet a aplikace pro konfiguraci bezkontaktních karet. Požaduje se, aby ke komunikaci s kartou byl využit software podle [2], hráčská aplikace běžela na Androidu a všechny ostatní aplikace běžely na Linuxu.

DOPORUČENÁ LITERATURA:

[1] Burda, K. Contactless Smart Card as a Cache for Geocaching. International Journal of Computer Science and Network Security, 2021, roč. 21, č. 7, s. 205-210. ISSN: 1738-7906.

[2] Vertaľ D.: Bezkontaktní mikropočítačová karta jako skrýš pro geokešing. [Diplomová práce] VUT v Brně, Brno. Dostupné na: https://bit.ly/3hgG7TR

Termín zadání: 6.2.2023 ***Termín odevzdání:*** 19.5.2023

Vedoucí práce: doc. Ing. Karel Burda, CSc.

doc. Ing. Jan Hajný, Ph.D.
předseda rady studijního programu

ABSTRAKT

Cílem práce je nastudovat problematiku běžného a elektronického geocachingu a provést implementaci navržené infrastruktury pro elektronický geocaching. Práce popisuje běžný geocaching, jeho historii a současnost a také co je samotná keš a co obsahuje. Dále je popsána infrastruktura pro elektronický geocaching, její části a komunikace mezi nimi. Práce dále uvádí návrh jednotlivých částí nové infrastruktury a jsou vybrány vhodné technologie pro jejich implementaci. Pro serverovou aplikaci byl vybrán framework Django a databáze PostgreSQL. Byl také proveden návrh jednotlivých modelů pro serverovou aplikaci. Pro klientskou aplikaci byl vybrán framework Angular. Pro klientskou aplikaci byly navrženy jednotlivé stránky a funkce těchto stránek. Následně jsou v práci otestovány již vytvořené aplikace pro chytrý telefon a čipovou kartu a je rozhodnuto, zda budou aplikace převzaty či nikoliv. V další části práce je provedena samotná implementace jednotlivých aplikací. U klientské aplikace je popsáno vytvoření komponentů stránek a jejich funkcí a také komunikace se serverovou aplikací. Část implementace serverové aplikace obsahuje vytvoření navržených modelů a databáze a také serializátorů modelů a přístupových bodů API (Application Programming Interface), díky kterým bude moci klientská a mobilní aplikace komunikovat se serverovou aplikací. V mobilní aplikaci byly vytvořeny jednotlivé aktivity a fragmenty a implementována komunikace se serverovou aplikací a čipovou kartou. Aplikace pro čipovou kartu byla převzata a jen mírně upravena. Práce dále popisuje testování jednotlivých aplikací a výsledky tohoto testování.

KLÍČOVÁ SLOVA

Geocaching, keš, Android, Linux, čipová karta, Dango, REST, API, Angular, certifkát

ABSTRACT

The aim of this work is to study conventional and electronic geocaching, and to implement the proposed infrastructure for electronic geocaching. The work describes convetional geocaching, it's history and present, as well as what is cache and what cache contains. next, the infrastructure for electronic geocaching, it's parts and communication between them is described. The work then describes the design of individual parts of the new infrastructure, and suitable technologies are selected for their implementation. The Django framework and PostgreSQL database were chosen for server appliaction.And a design for individual models for the server applicaation was made. The Angular framework was chosen for the client application. Design of individual pages and their functions was made for the client application. After the design, testing of already made applications was carried out, and it was decided, if the appliaction will be taken over or not. In the next part fo the work, the implementation of individual appliactions is made. The creation of page components and their functions, as well as communication with the server appliaction, is described for the client application. The server application implementation includes the creation of the designed models and database, as well as model serializers and API (Application Programming Interface) endpoints, that will allow the the client and mobile application to communicate with the server application. Individual activities and fragments were created for the mobile application, and communication with the server application and smart card was implemented.The smart card application has been taken over and slightly modified. The thesis further describes the testing of individual application and the results of this testing.

KEYWORDS

Geocaching, cache, Android, Linux, smart card, Django, REST, API, Angualr, certificate

ŠKRHÁK, Pavel. *Experimentální infrastruktura pro elektronický geokešing*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2023, 69 s. Diplomová práce. Vedoucí práce: doc. Ing. Karel Burda, CSc.

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Bc. Pavel Škrhák
VUT ID autora:	211583
Typ práce:	Diplomová práce
Akademický rok:	2022/23
Téma závěrečné práce:	Experimentální infrastruktura pro elektronický geokešing

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora*

*Autor podepisuje pouze v tištěné verzi.

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu doc. Ing. Karlu Burdovi CSc. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Obsah

Seznam obrázků

Seznam výpisů

Úvod

Geocaching je hra, kdy hráči hledají schránky, zvané keše, které jsou uschovány po celém světě [1]. Samotné keše mají různé velikosti a obsah. Keš nejčastěji obsahuje různé předměty, které mohou hráči vyměňovat, a logbook, do kterého zapíšou své jméno. Každá keš má své zeměpisné souřadnice a hráči je hledají pomocí GPS (Global Positioning System) [2]. Momentálně je po celém světě ukryto více než tři miliony různých keší a jejich počet se stále zvyšuje [1]. Tato diplomová práce se zabývá novou infrastrukturou pro geocaching, kdy keš již není fyzická schránka, ale bezkontaktní čipová karta.

Cílem diplomové práce je popis momentální problematiky jak běžného geocachingu, tak i elektronického geocachingu, a dále návrh jednotlivých částí infrastruktury, implementace navržených částí infrastruktury a otestování této infrastruktury

První kapitola práce se zabývá běžným geocachingem, jeho historií a současností. Na tuto kapitolu navazuje elektronický geocaching a především popis infrastruktury, která bude vytvořena v navazující kapitolách práce. Práce se dále zabývá certifikační autoritou a digitálními certifikáty, jak fungují a jaký je jejich formát. Ve čtvrté kapitole je proveden návrh jednotlivých částí infrastruktury a jsou vybrány vhodné programové prostředky pro implementaci jednotlivých částí. Tato kapitola se dále zabývá testováním již vytvořených aplikací, které implementují komunikaci mezi bezkontaktní čipovou kartou a mobilním telefonem s rozhraním NFC (Near Field Communication) [3]. Na základě testování je rozhodnuto, zda budou tyto aplikace použity v implementaci infrastruktury. Pátá kapitola je zaměřená na samotnou implementaci jednotlivých částí infrastruktury. Nejdříve jsou implementovány aplikace herního serveru, kde jako první proběhla implementace klientské aplikace a následně serverové aplikace, která využívá databázi pro ukládání dat. Dále byla vytvořena nová mobilní aplikace, která je schopná komunikovat s herním serverem a využívá kód pro komunikaci s bezkontaktní čipovou kartou, který byl převzat z původní aplikace [3]. Jako poslední byly provedeny drobné změny v původním appletu karty, které měly za cíl odstranit některé chyby. V šesté kapitole je provedeno testování vytvořené infrastruktury. Jsou otestovány funkce klientské i serverové aplikace, jako vytvoření putovního předmětu a objednávka karty a také test autentizace uživatele serverové aplikaci. Pro mobilní aplikaci proběhlo testování, kde bylo zjištěno, zda aplikace dostává data z herního serveru a zda aplikace otevře vytvořenou keš.

1 Geocaching

Geocaching je hra, která spojuje sport a turistiku. Celá hra je založena na hledání skrytých schránek zvaných keš (nazývána také keška, nebo anglicky cache či geocache) za použití systému GPS. O keši jsou známy pouze GPS souřadnice [2].

1.1 Historie

Geocaching vznikl v roce 2000, krátce poté, co byla odstraněna umělá odchylka ze systému GPS [4]. 3. května 2000 chtěl přesnost GPS systému otestovat Dave Ulmer tím, že uschová schránku do přírody a zveřejní její souřadnice. Tento nápad nazval *Great American* GPS Stash Hunt. Nálezce by měl tuto schránku nalézt pouze pomocí GPS přijímače [5].

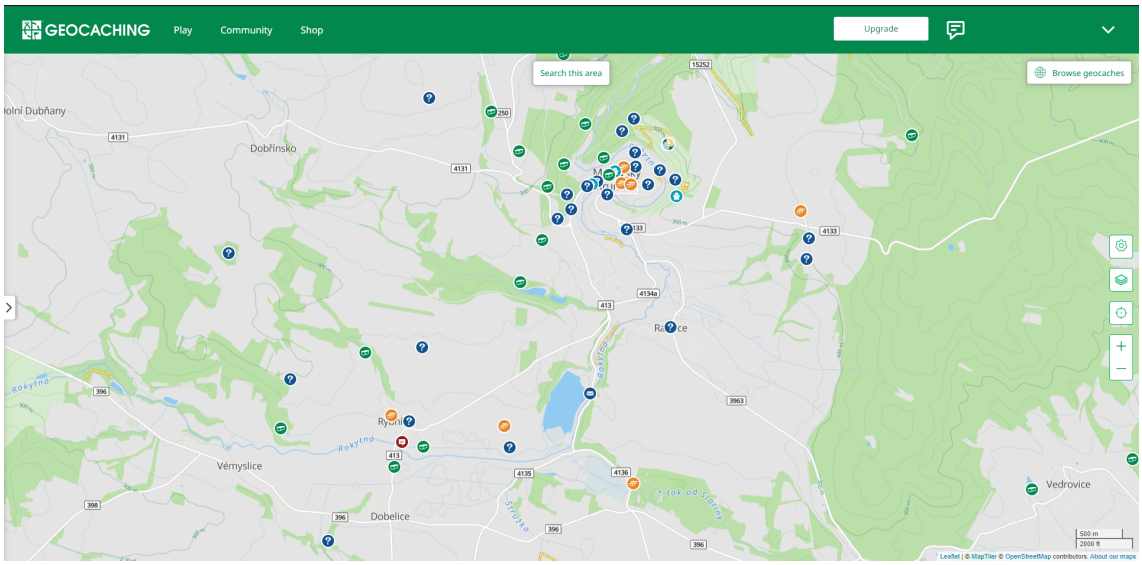
Následně Mike Teague, jenž našel Ulmerovu schránku jako první, založil seznam podobných schránek. Informace o nich shromažďoval z online příspěvků. V této komunitě se také začal používat současný název geocaching [5].

Ve stejném roce byly založeny i webové stránky pro geocaching, které jsou hlavními webovými stránkami celého geocachingu (www.geochaching.com). V době založení těchto stránek existovalo pouze 75 známých keší v celém světě. Ke konci roku 2000 byla založena organizace Groudspeak Inc., která momentálně provozuje stránky geocachingu [5].

1.2 Současnost

V této době je možné nalézt všechny informace o geocachingu na jeho stránkách. Na těchto stránkách je možné najít aplikaci pro geocaching, registraci uživatelů a nových keší, a také mapu se všemi kešemi. Mapa keší malé oblasti v České republice je viditelná na Obr.1.1. Dle organizace Groundspeak Inc. je ve světě více než tři miliony aktivních keší ve všech státech světa [6].

Geocaching nabízí mnoho typů her. Hlavním cílem je nalézt co největší množství keší. Motivací může být i navštívení cizích zemí a hledání keší v těchto zemích. Další hrou je velmi často také vyměňování putovních předmětů. Nejčastěji se hráči zapíšou do logbooku a následně vymění předmět uvnitř keše. Hráč zanechává v keši předmět stejné, nebo vyšší hodnoty, než který bere [7].



Obr. 1.1: Mapa malé oblasti v České republice na webu www.geocaching.com.

1.3 Keš

Hráč nemusí pouze hledat vytvořené keše, ale může také založit svou vlastní. Hráči často keš vytváří po delší době hraní, kdy sbírají zkušenosti jak vytvořit zajímavou keš [8].

V současnosti je možné vytvořit 16 typů keší. Jedná se jak o běžné keše, kde je pouze logbook a předmět pro výměnu, tak také o keše, které jsou spojeny s různými typy her. U mystery keší je například nejprve nutné vyřešit různě složité hádanky pro získání správných souřadnic. Krom těchto možností existují další čtyři typy keší, které ale už není možné vytvořit [9].

Mezi nejznámější typy keší patří [9]:

- Tradiční keš: Originální typ keše, obsahuje alespoň logbook.
- Mystery keš: Často obsahuje různě složité hádanky k odhalení polohy keše.
- Multi-Cache: Skládá se z několika zastávek, kde každá zastávka je na jiném místě a obsahuje informace o další zastávce (souřadnice). Poslední zastávka je keš s logbookem.
- EarthCache: Často na zajímavých geologických lokalitách, stránky dané EarthCache obsahují krom souřadnic také poučné informace o lokalitě.
- Letterbox Hybrid: Používá místo souřadnic nápovědy jak keše najít.
- Event Cache: Pro různé plánované akce, kde se scházejí hráči geocachingu. Je specifikováno místo a čas události, po skončení je keš archivována.

2 Elektronický geocaching

Už nyní je možné říct, že geocaching je v jisté formě z části elektronický. Většina hráčů používá chytré telefony pro navigaci ke keši a krom zápisu do fyzického logbooku v keši často používají i zápis do logu v oficiální aplikaci. V některých zemích se však začínají objevovat elektronické logbooky. Tyto logbooky používají WiFi modul, díky kterému se můžou hráči připojit a zapsat do logbooku. Napájení pro toto zařízení je ve většině případů řešeno pomocí USB (Universal Serial Bus) kabelu, který hráč připojí ke svému chytrému telefonu [10].

Hlavním zdrojem této práce je elektronický článek, který se zabývá novou architekturou pro elektronický geocaching. Tato architektura využívá bezkontaktní čipovou kartu jako keš a hráčskou aplikaci na chytrém mobilním telefonu s rozhraním NFC pro komunikaci s touto kartou [11].

2.1 Popis architektury

Celá architektura se skládá z pěti hlavních komponentů. Prvním komponentem je certifikační autorita. Certifikační autorita je zodpovědná za generování certifikátu pro ostatní komponenty. Tyto certifikáty jsou podepsány certifikační autoritou [11].

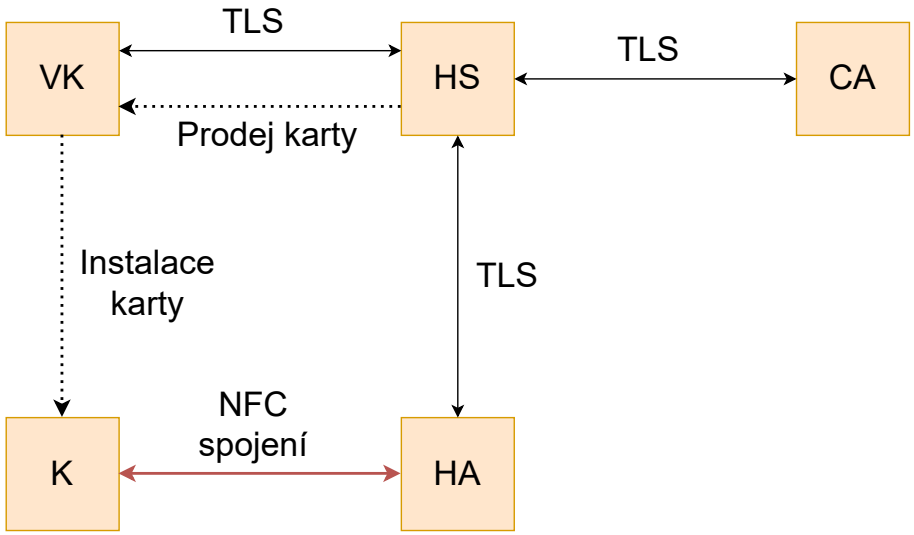
Herní server obsahuje záznamy o všech hráčích a keších, má také vygenerován veřejný a soukromý klíč. Soukromý klíč herního serveru je použit pro autentizaci serveru pro ostatní komponenty. Veřejný klíč je použit ostatními komponenty pro ověření identity herního serveru. Herní server dále slouží jako registrační autorita, které hráči podávají žádosti o certifikáty. Herní server tyto žádosti zpracovává a předává certifikační autoritě. Certifikáty vygenerované certifikační autoritou jsou předány zpět na herní server, který je následně dále odesílá hráčům. Další funkcí herního serveru je zpracovávání a ukládání herních výsledků hráčů. Poslední funkcí herního serveru je objednání keše [11].

Hráči jsou v systému reprezentováni ne jako osoby, ale jako jejich hráčské aplikace. Tato aplikace je stažena z herního serveru. Aplikace je nainstalována na chytrý telefon, který také obsahuje NFC rozhraní pro komunikaci s bezkontaktními čipovými kartami. Součástí aplikace je také certifikát obsahující veřejný klíč certifikační autority. Tento klíč je použit aplikací pro autentizaci ostatních komponentů. Aplikace následně vygeneruje veřejný a soukromý klíč hráče, resp. hráčské aplikace. Hráč je následně zaregistrován na hráčském serveru, kde zadá svoji jedinečnou přezdívku, a je mu vygenerován certifikát pro jeho veřejný klíč. Tento certifikát je vygenerován certifikační autoritou a je předán přes herní server [11].

Dalším komponentem je vlastník keše. Tato osoba na herním serveru provede registraci a objednání karty. Specializovaná osoba následně vygeneruje veřejný a

soukromý klíč karty a prostřednictvím certifikační autority získá certifikát karty. Specializovaná osoba poté provede zápis potřebných dat na kartu. Takto vytvořená karta je fyzicky odeslána osobě, která ji registrovala. Vlastník keše tuto kartu umístí na její místo a označí keš jako aktivovanou [11].

Posledním komponentem je samotná čipová karta. Tato karta obsahuje jedinečný identifikátor karty, soukromý klíč karty, certifikát veřejného klíče karty a certifikát veřejného klíče certifikační autority. Karta je schopná komunikovat s hráčskou aplikací pomocí komunikačního rozhraní chytrého telefonu. Schéma celé architektury je zobrazeno na Obr.2.1 [11].



Obr. 2.1: Architektura systému [11].

2.2 Popis komunikace

Krom samotné architektury je také určeno, jakým způsobem jednotlivé komponenty komunikují. Komunikace mezi herním serverem, vlastníkem keše, hráčskou aplikací a certifikační autoritou je zabezpečena pomocí protokolu TLS (Transport Layer Security) [11]. Tento protokol zajišťuje autentizaci komunikujících stran, důvěrnost přenášených dat a integritu těchto dat [12]. Pro protokol TLS je možné použít certifikáty generované certifikační autoritou.

I když jsou certifikační autorita a herní server jiné komponenty, je možné je oba provozovat na stejném zařízení. V tomto případě by nebylo třeba mít TLS spojení mezi certifikační autoritou a herním serverem. Komunikace těchto dvou komponentů tedy závisí na umístění certifikační autority.

Komunikace mezi herním serverem a vlastníkem keše probíhá pomocí webového prohlížeče přes protokol HTTPS (Hypertext Transfer Protocol Secure). Tímto způsobem může vlastník keše objednat a spravovat své keše. U hráčské aplikace je nejpravděpodobnější komunikace pomocí HTTP požadavků na herní server [11].

2.2.1 Komunikace karty s aplikací

Karta s hráčskou aplikaci komunikuje přes rozhraní NFC [11]. NFC přenáší energii a data pomocí indukce magnetického pole. Komunikace mezi zařízeními může probíhat pouze na velmi krátkou vzdálenost [13].

Specifikace v příloze obsahuje požadavky na běh protokolu. Pro přenos hodnot bude použit formát AVP (Attribute–Value Pair). Každá hodnota je přenášena jako trojice, kde první bajt udává typ hodnoty, druhý bajt celkovou délku trojice a zbylé bajty přenášenou hodnotu. Dále je uvedeno, že pro podpis bude použit protokol ECDSA (Elliptic Curve Digital Signature Algorithm) 192 bitů, kde délka náhodných čísel je minimálně 32 bitů.

Průběh samotného protokolu začíná navázáním spojení mezi chytrým telefonem s hráčskou aplikací a kartou. Po navázání spojení mezi aplikací a kartou je odeslána zpráva *Start* hráčskou aplikací. Karta na tuto zprávu odpoví dvojicí R_S, S, kde R_S je náhodné číslo a S je jedinečné číslo karty. Po přijetí této dvojice hráčská aplikace zkontroluje, zda tato keš byla již hráčem nalezena. Pokud již byla nalezena, je komunikace přerušena zprávou *Stop*, pokud nebyla nalezena, pokračuje komunikace dál [11].

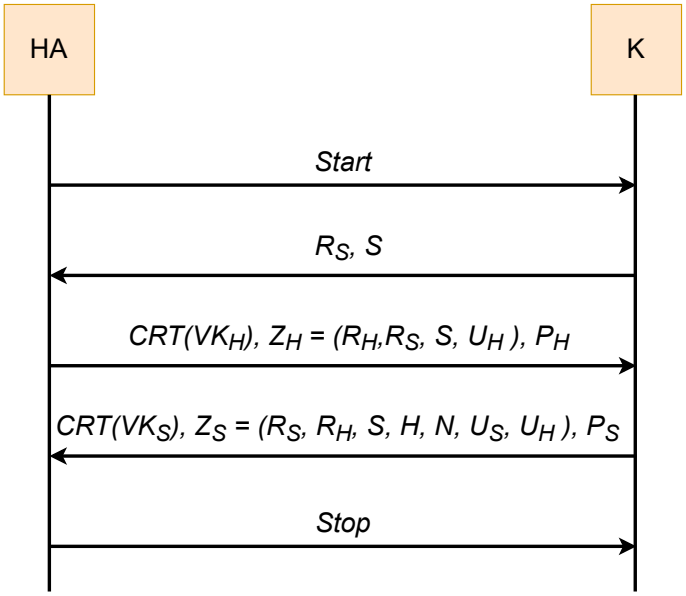
Hráčská aplikace následně odešle certifikát hráče, zprávu hráče Z_H (viz rovnice 2.1) a podpis této zprávy P_H kartě. Karta ověří platnost certifikátu pomocí veřejného klíče certifikační autority a pokud je platný, ověří pomocí veřejného klíče hráče podpis zprávy. Zpráva hráče obsahuje náhodné číslo hráče R_H, náhodné číslo karty R_S, jedinečné číslo karty S a číslo putovního předmětu U_H [11].

$$Z_H = (R_H, R_S, S, U_H) \tag{2.1}$$

Karta ověří, že se náhodné a jedinečné číslo karty shodují s číslem které bylo odesláno a uloží náhodné číslo hráče. Pokud je hodnota čísla putovního předmětu nula, nemá hráč žádný putovní předmět. Karta poté odešle trojici, která obsahuje certifikát karty, zprávu Z_S a její podpis P_S. Zpráva Z_S obsahuje obsah zprávy Z_H, a navíc také číslo hráče H (toto číslo je zjištěno z certifikátu hráče), číslo N, které říká kolikátým nálezcem hráč je a číslo U_S, které obsahuje číslo putovního předmětu na kartě viz rovnice 2.2 [11].

$$Z_S = (R_S, R_H, S, H, N, U_S, U_H) \tag{2.2}$$

Aplikace nejdříve zkontroluje, zda se shodují náhodná čísla R_S a R_H, a dále jedinečné číslo karty S a číslo hráče H. Zpráva funguje jako potvrzení že hráč je N-tý nálezce keše, a také že proběhl přenos putovních předmětů. Po této zprávě aplikace ukončí přenos pomocí zprávy *Stop*. Karta po přijetí zprávy zvýší hodnotu čísla N. Schéma celé komunikace je vyobrazeno na Obr.2.2 [11].



Obr. 2.2: Schéma komunikace karty a hráčské aplikace [11].

Aplikace po přijetí a ověření zprávy Z_S kontaktuje herní server. Aplikace předá trojici obsahující certifikát, zprávu karty a podpis zprávy serveru. Server zkontroluje platnost certifikátu, a upraví záznam o keši a hráči [11].

2.3 Bezpečnost architektury

Bezpečnost komunikace komponentů s herním serverem je založena na použití protokolu TLS. Tento protokol je považován za bezpečný. Bezpečnost při komunikaci mezi bezkontaktní čipovou kartou a hráčskou aplikací je založena na několika faktorech. Prvním z nich jsou vygenerovaná náhodná čísla R_S a R_H. Tyto čísla jsou jiná pro každý běh aplikace. Tyto čísla a jejich kontrola při každém kroku zajišťují, že případný útočník nemůže použít zprávy z předchozího přenosu. Druhým faktorem jsou podpisy zpráv, které zajišťují autenticitu přenášených zpráv, a zároveň také identitu komunikujících stran [11].

3 Certifikační autorita

Certifikační autorita je entita, která vydává digitální certifikáty. Této entitě je důvěřováno a vzniká tedy i důvěra pro certifikáty vydané touto entitou. Certifikáty vydané certifikační autoritou mají více využití (např. pro digitální podpis), nejznámější je však použití digitálních certifikátů pro navázání zabezpečeného spojení mezi serverem a webovým prohlížečem uživatele [14].

Certifikační autorita je jednou z hlavních částí infrastruktury veřejných klíčů (anglicky Public Key Infrastructure, nebo ve zkratce PKI). Hlavním cílem infrastruktury veřejných klíčů je bezpečná distribuce veřejných klíčů a digitálních certifikátů. Mezi další hlavní části patří digitální certifikáty, registrační autorita a databáze certifikátů [15].

Registrační autorita ověřuje identitu žadatele o certifikát a také zda splňuje podmínky pro udělení certifikátu. Tuto funkci provádí pro danou certifikační autoritu s jejím svolením [15].

Databáze certifikátů se používá pro uchování informací o vydaných certifikátech a také seznamu zneplatněných certifikátů. Tyto certifikáty a seznam neplatných certifikátů je veřejně dostupný [15].

3.1 Digitální certifikát

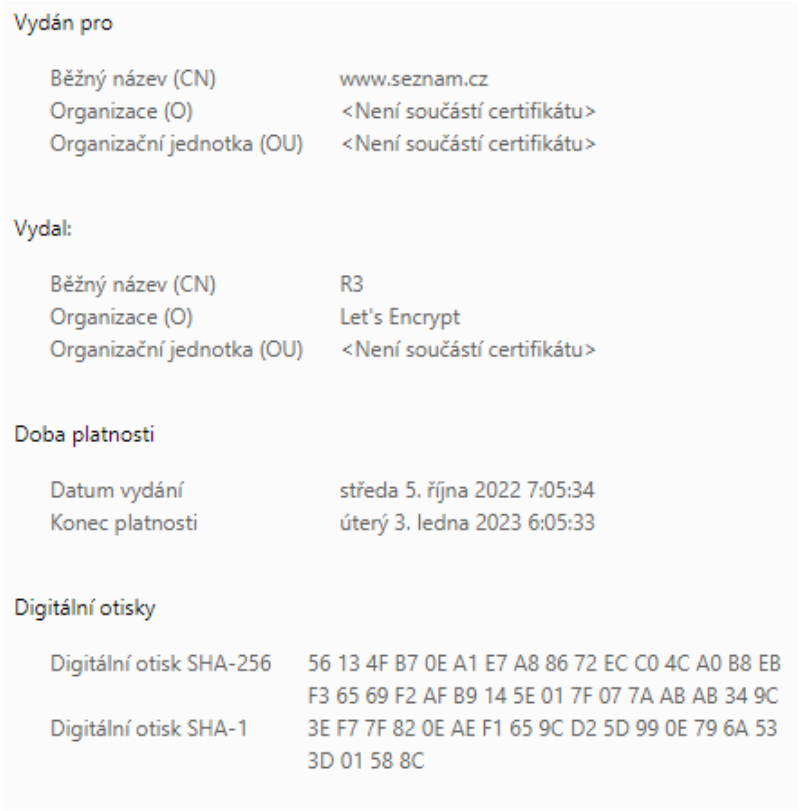
Digitální certifikát je prostředek, díky kterému je možné ověřit identitu entity, které byl udělen. Krom ověření identity se také často používá pro šifrování komunikace, či pro zaručení integrity pomocí digitálního podpisu [14].

Běžně používané jsou certifikáty dle standardu X.509. Tento standard je popsán v RFC 5280 [16].

Tento standard udává formát digitálních certifikátů pro použití v infrastruktuře veřejných klíčů. Hlavní částí certifikátu jsou pole obsahující informace o entitě, které byl certifikát vydán, a také o certifikační autoritě, která certifikát vydala. Příklad obecných informací o certifikátu je na Obr.3.1 [16].

- Verze - udává verzi certifikátu
- Sériové číslo - sériové číslo certifikátu udělené certifikační autoritou
- Algoritmus podpisu - obsahuje informace o algoritmu použitém pro podpis certifikátu certifikační autoritou
- Vydavatel certifikátu - identifikace certifikační autority, která certifikát vydala a podepsala
- Datum platnosti - udává časové rozmezí platnosti certifikátu
- Subjekt - informace o entitě, které byl certifikát vydán

- Veřejný klíč subjektu - obsahuje veřejný klíč a také informace o algoritmu, pro který má být použit
- Rozšíření - Další rozšiřující pole certifikátu
- Digitální podpis - podpis certifikátu certifikační autoritou



Obr. 3.1: Obecné informace o certifikátu pro www.seznam.cz.

Jelikož je možné využít certifikáty ve formátu X.509 pro více funkcí, je často přidáváno pole, které určuje použití certifikátu. Díky tomuto poli je možné omezit použití certifikátu. Toto pole je jedno z několika volitelných rozšiřujících polí, které lze dle standardu X.509 využít [16].

4 Návrh infrastruktury

4.1 Herní server

V rámci práce budou jak certifikační autorita, tak i herní server instalovány na stejném zařízení. Díky tomuto není třeba vytvářet zabezpečené spojení mezi dvěma zařízeními. Jako zařízení je vybrán linuxový operační systém Ubuntu. Vzhledem k funkci zařízení bude použit Ubuntu server. Ubuntu server je dostačující pro funkce certifikační autority i herního serveru, zároveň má nižší nároky než běžný operační systém Ubuntu.

4.1.1 Certifikační autorita

Pro vytvoření certifikační autority na zařízení je možné použít OpenSSL. Tento software je chopen generovat kryptografické klíče a také digitální certifikáty.

Na zařízení bude nutné vytvořit adresáře pro uchovávání certifikátů a klíčů. Hlavní částí bude adresář pro vydané certifikáty. Další z potřebných adresářů bude adresář pro seznam zneplatněných certifikátů. Dále bude třeba adresář pro uchovávání soukromých klíčů. Jako poslední bude vhodné vytvořit adresář pro žádosti o certifikáty.

4.1.2 Serverová aplikace

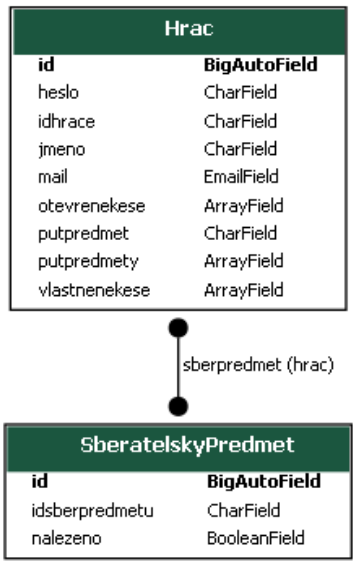
Pro databázi serveru bude využit databázový systém PostgreSQL. Důvodem je jednoduchost použití tohoto systému, převážně díky aplikaci pgAdmin, která umožňuje jednoduchou správu databází. Tento program disponuje přehledným uživatelským rozhraním. Dále se také jedná o jeden z nejpoužívanějších systémů pro SQL (Structured Query Language) databáze. Tato databáze bude uchovávat informace jak o kartách, tak o jednotlivých hráčích, a také o putovních předmětech. Jelikož budou o hráčích, kartách a putovních předmětech uchovávány rozdílné informace, je nutné navrhnout pro všechny tři objekty různé tabulky databáze. Pro návrh tabulek je nejprve třeba rozlišit, jaké informace se budou o objektech uchovávat.

Nejvíce informací bude nutné uchovávat o hráčích samotných. Krom běžných informací jako jsou jméno, heslo a e-mailová adresa, bude pro každého hráče vytvořen seznam otevřených keší a také seznam keší které vlastní. Dále je třeba uchovat informaci o aktuálním putovním předmětu, pokud nějaký vlastní, případně informaci, že hráč nevlastní žádný putovní předmět. Jelikož hráči budou mít možnost putovní předměty vytvořit, tak musí být uchován také seznam vytvořených putovních předmětů pro každého hráče. Poslední informací o hráči, kterou bude nutné ukládat, jsou jeho sběratelské předměty.

O samotné keši bude třeba ukládat menší množství informací. Nejdůležitější informace jsou její identifikační číslo, název, poloha a počet nálezů. Dále bude uložena informace o putovním předmětu uvnitř keše a také o jejím vlastníkovi.

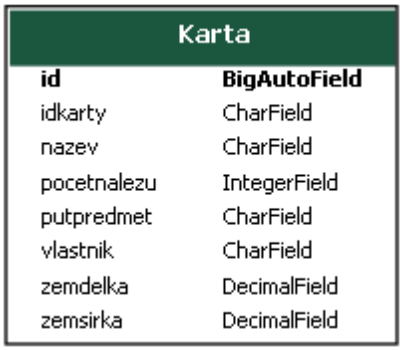
Posledním prvkem, o kterém bude nutné ukládat informace, je putovní předmět. O putovním předmětu bude uloženo jeho identifikační číslo, vlastník, cesta k souboru a také seznam všech keší, ve kterých se nacházel. Poslední informací o putovním předmětu bude identifikační číslo momentálního nosiče putovního předmětu, a také zda tento nosič je hráčská aplikace, nebo keš. Databáze samotná není vhodná k ukládání obrázků, které budou představovat putovní předměty. Z tohoto důvodu bude obrázek uložen na serveru a v databázi uložena cesta k tomuto obrázku. Seznam keší bude použit pro tvorbu mapy pohybu putovního předmětu.

Pro práci s databází bude použit framework Django. Hlavním důvodem použití je předchozí praxe s tímto frameworkem. Framework Django je běžně používaný pro tvorbu webových aplikací. Tento framework také velmi zjednodušuje práci s SQL databázemi. Django využívá tzv. modely, které uživatel vytvoří. Tyto modely jsou v podstatě šablona, podle které jsou vytvořeny jednotlivé tabulky v databázi. Při návrhu modelu hráče ve frameworku Django je nutné vyřešit problém se sběratelskými předměty. O sběratelských předmětech jsou známy dvě hodnoty. První z nich je jejich identifikační číslo a druhá obsahuje informaci, zda již byl hráčem nalezen. Framework Django nepodporuje žádný datový typ, který by umožňoval vložit více různých datových typů do jednoho pole. Je tedy nutné vytvořit nový model pro sběratelské předměty a tento model následně použít v modelu hráče, viz Obr:4.1. Framework Django umožňuje použití pole. Seznamy vytvořených putovních předmětů, otevřených a vlastněných keší, budou tedy uloženy v polích [17].



Obr. 4.1: Návrh modelu hráče.

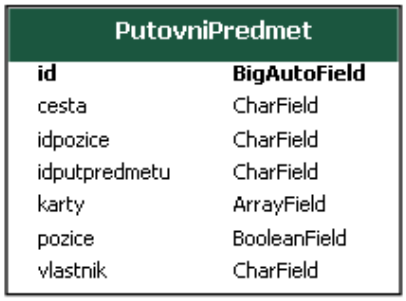
Návrh modelů pro kartu je vcelku jednoduchý. Jediný problém nastává u zeměpisných souřadnic, jelikož framework Django nepodporuje žádný datový typ určený k jejich ukládání. Souřadnice budou tedy uloženy ve dvou polích, zeměpisná délka a zeměpisná šířka, Obr:4.2 [18].



Obr. 4.2: Návrh modelu keše.

Návrh modelu putovních předmětů probíhá podobně jako u předchozích. Je třeba rozhodnout, jaké typy polí budou použity. Podobně jako u návrhu modelu hráče bude vhodné pro uložení seznamu keší, kde se předmět nacházel, použití pole viz Obr:4.3.

Krom správy databáze musí back-end také komunikovat s front-endem a hlavně s mobilní aplikací. Pro komunikaci s mobilní aplikací bude využito API (Application Programming Interface). I když je možné ve frameworku Django vytvořit API, je



Obr. 4.3: Návrh modelu putovního předmětu.

vhodnější využít Django REST framework. Tento framework umožňuje mnohem jednodušší tvorbu REST (Representational State Transfer) API v běžných Django aplikacích. Pomocí REST API je možné provádět různé operace. V případě mobilní aplikace i front-endu bude nejdůležitější získávání dat z databáze a úprava dat v databázi [19].

Jako první je vhodné vyřešit registraci uživatelů a přihlašování existujících uživatelů. Pro registraci uživatelů bude nutné přenést jejich přezdívku, e-mailovou adresu a heslo a tyto informace uložit do databáze. Při přihlašování uživatel zadá pouze přezdívku a heslo. Framework Django již obsahuje funkci pro autentizaci uživatelů. Po bezpečném přenesení přihlašovacích údajů z front-endu do Django aplikace proběhne kontrola údajů. Pokud jsou údaje správné, Django odešle odpověď zpět na front-end, který dále bude pracovat na základě úspěšné či neúspěšné autentizace.

REST API v Django REST frameworku přijímá data ve formátu JSON JavaScript Object Notation. I když databáze zpracovává data v jiném formátu, není třeba řešit složité převádění dat z API. Django REST framework obsahuje tzv. serializéry, které data automaticky převedou na potřebný formát. API bude nutné vytvořit pro všechny modely, krom modelu pro sběratelské předměty. Příklad vytvořeného API je znázorněn na Obr:4.4. Data přenesená z API následně zpracuje buď front-end, nebo mobilní aplikace [19].

```
HTTP 201 Created
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "jmeno": "hrac2",
    "mail": "hrac2@hrac.com",
    "heslo": "heslo2",
    "idhrace": "0002",
    "otevrenekese": [],
    "putpredmet": "",
    "putpredmety": [],
    "sberpredmet": [],
    "vlastnenekese": []
}
```

Obr. 4.4: Návrh API pro model hráče.

4.1.3 Klientská aplikace

Pro front-end bude použit framework Angular. Důvodem použití je přehlednost tohoto frameworku při práci sním. Angular umožňuje aplikaci rozdělit do menších komponentů, které je možné použít kdekoliv aplikaci. Angular je jedním z nejpoužívanějších frameworků pro tvorbu webových klientských aplikací (aplikací běžících ve webovém prohlížeči). Angular je založen na programovacím jazyku TypeScript [24]. TypeScript byl vyvinut jako rozšíření JavaScript, který často nebyl vhodný pro rozsáhlé aplikace. TypeScript přidává možnost deklarovat typy proměnných a také přidává například možnost vytváření tříd [21].

Jelikož bude ve hře více hráčů, kteří se můžou přihlásit do svých herních účtů, bude potřeba vytvořit stránku pro registraci a přihlašování hráčů. Stránka bude nabízet možnost buď registrace, nebo přihlášení. Z této stránky bude hráč přesměrován na hlavní stránku, kde budou vypsány jeho statistiky. Bude zde jeho přezdívka, celkové množství nalezených keší a putovní předmět. Z této hlavní stránky se poté bude moci hráč přesunout na jednu z několika dalších stránek. Bude se jednat o informace o keších, putovních předmětech, sběratelských předmětech a také o mapu obsahující aktivní keše.

Stránka obsahující informace o keších, které hráč vlastní, bude obsahovat informace o celkovém počtu nalezení a putovním předmětu uvnitř keše. Bude se zde také nacházet možnost zakoupení keše.

Další stránkou budou putovní předměty hráče. Zde budou informace o aktuálním putovním předmětu a také o předmětech, které hráč vytvořil. Opět zde bude

možnost vytvoření nového putovního předmětu, pokud hráč žádný momentálně nevlastní. Součástí bude i mapa, která bude znázorňovat pohyb putovních předmětů vytvořených hráčem.

Stránka s informacemi o sběratelských předmětech bude zobrazovat doposud získané sběratelské předměty. Stránka také zobrazí počet doposud získaných a nezískaných sběratelských předmětů.

Komunikace klienta s webovou aplikací bude zabezpečena pomocí TLS. Angular, podobně jako i většina ostatních frameworků, podporuje toto zabezpečení. Pro webovou aplikaci bude nutné vygenerovat certifikát, který bude při komunikaci použit. Komunikace s back-endem bude zabezpečena pomocí JWT (JSON Web Token) [24].

JWT je standart RFC 7519. Definuje, jak bezpečně přenášet informace mezi stranami pomocí JSON objektů. Tyto informace jsou podepsány. JWT se nejčastěji používá pro autorizaci a pro přenos informací. JWT se skládá ze tří částí - hlavičky, payloadu a podpisu. Hlavička obsahuje informace o typu tokenu a algoritmu použitém pro digitální podpis. Payload obsahuje tzv. tvrzení. Jedná se o tvrzení převážně o uživateli. Tvrzení se dělí na tři typy - registrované, veřejné a soukromé. Poslední část tokenu je digitální podpis. Jedná se o podpis hlavičky i payloadu. Obě tyto části jsou zakódovány a je k nim přidáno tajné heslo. Tyto data jsou poté podepsány a podpis je přidán to tokenu. [22]

4.2 Android aplikace

Hráčská aplikace bude běžet na mobilních telefonech s operačním systémem Android. Podmínkou pro správný běh aplikace je také rozhraní NFC, které bude použito pro komunikaci s bezkontaktní čipovou kartou. Hráčská aplikace bude zobrazovat podobné informace jako webové stránky. Bude se jednat o informace o hráči, informace o putovním předmětu a informace o sběratelských předmětech. Aplikace dále bude disponovat mapou, na které budou zobrazeny keše v okolí hráče. Aplikace bude schopná tyto data získat z herního serveru.

Komunikace s herním serverem bude probíhat pomocí HTTP požadavků. Před přístupem k REST API proběhne přihlášení hráče k hernímu serveru. Po přihlášení mu bude vygenerován JWT, díky kterému bude mít aplikace přístup k API. Aplikace dále získá potřebná data pro zobrazení všech informací. Po nalezení keše aplikace naopak data odešle na herní server, kde se uloží do databáze.

Krom komunikace s databází bude aplikace také komunikovat s keší, tedy bezkontaktní čipovou kartou. Komunikace bude probíhat přes NFC pomocí protokolu popsaného ve druhé kapitole této práce.

Jako základ bude použita vytvořená aplikace z diplomové práce *Bezkontaktní mikropočítačová karta jako skrýš pro geokešing* [3]. Aplikace pro chytrý telefon obsahuje v podstatě vše potřebné pro komunikaci s bezkontaktní čipovou kartou. Aplikace také umožňuje výběr typu hry a funkční výměnu předmětů. Aplikace jako taková však bude potřebovat několik úprav. Grafické uživatelské rozhraní aplikace je vytvořeno pouze pro výběr typu hry a zobrazení výsledku běhu protokolu. Bude nutné jej změnit, aby bylo možné zobrazit i ostatní informace, které bude aplikace mít, a také aby bylo možné vybrat více funkcí aplikace. Dále bude nutné přidat funkce pro komunikaci s herním serverem. Bude také potřeba vytvořit funkce pro dané hry, jako je například doplnění skládačky. I když aplikace tyto hry umožňuje vybrat, nejsou plně implementovány a jsou jen připraveny jejich metody. Ve výpisu 4.1 je zobrazena část funkce pro výběr různých her. Tato část funkce obsahuje situaci, kdy hráč vybral hru skládačka. Funkce momentálně pouze zobrazí obrázek skládačky [3].

Výpis 4.1: Část funkce pro výběr typu hry v aplikaci.

```
1         if (stringGame.equals("Skladačka")){
2             try {
3                 showImage("Skladačka", view);
4                 popupGame.dismiss();
5             } catch (Exception e) {
6                 e.printStackTrace();
7             }
8         }
```

4.3 Applet pro bezkontaktní čipovou kartu

Jako bezkontaktní čipová karta bude použita Java Card verze 2.2.2. Od verze Java Card se odvíjí potřebné vývojové prostředí. Java Card je omezenou verzí programovacího jazyka Java. Toto omezení je znát především na podporovaných datových typech [23].

Hlavní funkcí, kterou bude applet karty provádět, bude komunikace s hráčskou aplikací. Karta bude dále uchovávat informace o putovním předmětu uvnitř karty a také o počtu nalezení. Výše zmíněná diplomová práce obsahuje krom aplikace na chytrý telefon i applet pro bezkontaktní čipovou kartu [3]. V appletu jsou data uložena jako pole bajtů, viz 4.2. Výhodou tohoto formátu je snadné převedení do APDU (Application Protocol Data Unit). APDU je datová jednotka pro přenos dat mezi čipovou kartou a čtečkou čipových karet . Vytvořený applet obsahuje všechny potřebné funkce pro komunikaci s kartou a běh protokolu popsaného ve druhé kapitole [3][23].

Výpis 4.2: Příklad uložení hodnoty v poli bajtů.

```
1 final static byte[] bSKs = new byte[]{(byte) 0x38, (byte) 0
    x47, (byte) 0x64, (byte) 0x78, (byte) 0xE1, (byte) 0x94, (
    byte) 0x5F, (byte) 0x4C, (byte) 0xBF, (byte) 0x7F, (byte)
    0x82, (byte) 0xF8, (byte) 0x2A, (byte) 0x5B, (byte) 0x28,
    (byte) 0xC2, (byte) 0x36, (byte) 0x53, (byte) 0xE0, (byte)
     0x3E, (byte) 0x6E, (byte) 0x72, (byte) 0x2D, (byte) 0x3A
    };
```

Applet používá k podpisu a ověření podpisů zpráv ECDSA. K běhu tohoto algoritmu je nutné mít parametry dané křivky. Applet z diplomové práce používá křivku SECP192K1. Parametry této křivky jsou uloženy samostatně ve vlastní třídě. Krom této křivky obsahuje tato třída také parametry křivky SECP256K1. Tato třída je poté importována do hlavní třídy appletu. Jelikož je možné použít obě křivky, obsahuje applet přepínač pro výběr křivky k použití [3].

Součástí práce bude také software pro tvorbu a úpravu appletů pro bezkontaktní čipové karty. Pro tvorbu a úpravu samotných karet jsou dvě možnosti. Buď budou úpravy prováděny ručně v libovolném IDE (Integrated Development Environment), nebo bude pomocí skriptu upravován samotný soubor fungujícího appletu. Pro ruční úpravu je vhodné vývojové prostředí Eclipse, které umožňuje instalaci doplňků pro práci s Java Card. Jelikož se u jednotlivých karet bude měnit především certifikát, není ruční metoda úprav vhodná. Pro vytváření karet bude tedy implementován skript, který upraví potřebná data a pomocí nástrojů obsažených v balíku nástrojů k práci s Java Card převede a vygeneruje potřebné soubory pro instalaci na kartu. Skript samotný pouze převede potřebná data na pole bajtů a tato pole vloží na daná místa v hlavním souboru appletu [23].

4.4 Testování aplikací

Testování proběhlo pomocí aplikací přiložených k původní diplomové práci. Pro testování bylo použito pět bezkontaktních čipových karet a chytrý telefon s rozhraním NFC.

4.4.1 Příprava testování

Na dvě z pěti karet byl nainstalován původní applet, který byl přiložen k původní diplomové práci [3]. Měly tedy stejné identifikační číslo karty a putovní předmět. Applet pro ostatní karty byl upraven pro účely testování. Třetí karta obsahovala putovní předmět s jiným identifikačním číslem. Čtvrtá karta obsahovala jiné identifikační číslo karty. Poslední karta obsahovala identifikační číslo, které bylo uvedeno v seznamu nalezených keší v kódu mobilní aplikace 4.3.

Výpis 4.3: Pole otevřených keší v mobilní aplikaci.

```
1 IDs = new ArrayList<String>();
2         IDs.add("GC2222222222");
3         IDs.add("GC3333333333");
4         IDs.add("GC4444444444");
5         IDs.add("GC5555555555");
```

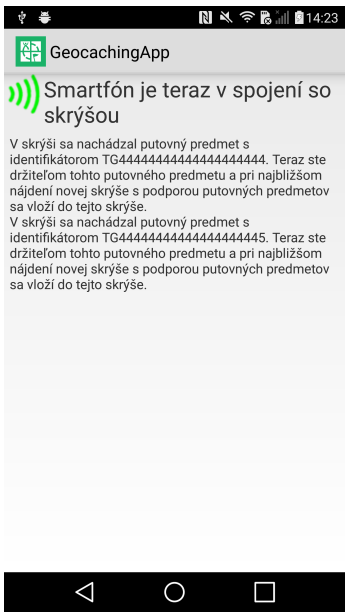
Karty, respektive keše, byly otevřeny v tomto pořadí. Nejdříve byla nalezena jedna z keší s původním appletem, následně keš s jiným putovním předmětem, poté keš s jiným identifikačním číslem a dále druhá keš s původním appletem. Jako poslední byla nalezena keš s identifikačním číslem, které je uvedeno jako již otevřené v kódu mobilní aplikace.

4.4.2 Výsledky testování

V průběhu testování byly odhaleny některé nedostatky původní mobilní aplikace. Pravděpodobně největším problémem je rozeznání již nalezených keší aplikací. Při nalezení dvou totožných keší jsou obě keše úspěšně otevřeny. Při pokusu o znovunalezení stejné keše však aplikace hlásí chybu. Jelikož se však jedná o základní chybovou hlášku, která je hlášena při potížích s kartou, není možné zde rozpoznat, zda se keš uložila do seznamu mobilní aplikace, nebo zda přestala fungovat keš samotná. Ostatní chyby jsou pouze menší a jsou závislé spíše na chybách v samotném grafickém uživatelském prostředí mobilní aplikace.

Jediné dvě další chyby, které bude třeba opravit, se týkají právě grafického uživatelského rozhraní mobilní aplikace. Bez přiložené karty aplikace správně zobrazí, že k telefonu není přiložena žádná karta. Stejně tak při přiložení karty je tato karta

detekována a je zobrazeno oznámení, že je přiložena karta. Po odejmutí karty se však tato hláška nezmění a zůstane pouze hláška, že je přiložena karta. Tato hláška se změní pouze po stisknutí tlačítka pro otevření keše. Druhý problém nastal při otevření dvou keší s jiným putovním předmětem. Po otevření keše se vypíše hláška, která říká který putovní předmět se v keši nacházel. Hláška dále říká, že tento předmět byl přesunut do mobilní aplikace, proběhla tedy výměna putovního předmětu. Při otevření keše s jiným sběratelským předmětem se však tato hláška nepřepíše, ale doplní se druhá hláška s informacemi o druhém putovním předmětu. Po zobrazení této hlášky zmizí tlačítko pro otevírání keší 4.5.



Obr. 4.5: Aplikace po otevření druhé keše.

Vzhledem k potřebným úpravám a přidání funkcí nebude mobilní aplikace převzata v celém rozsahu. Z aplikace bude převzat kód pro komunikaci s bezkontaktní čipovou kartou a běh protokolu pro výměnu informací mezi kartou a chytrým telefonem. Dále bude převzat kód pro výběr různých typů her. Tato část kódu bude však dále upravena a budou doplněny funkce pro běh samotných her. Bude tedy vytvořena nová aplikace, která bude implementovat části vytvořeného kódu. Hlavní částí bude vytvoření nového uživatelského rozhraní a přidání funkcí pro komunikaci s herním serverem.

Jelikož je applet pro kartu funkční, bude celý kód převzat. V kódu budou provedeny malé úpravy. Hlavní část těchto úprav bude přesun parametrů křivky do hlavního souboru appletu. Další úpravou bude odstranění parametrů křivky SPEC256K1 a také odstranění přepínače pro výběr mezi nimi. Budou také přidány kódy pro různé typy chyb v běhu protokolu. Applet momentálně používá základní chybovou hlášku. S touto hláškou však není možné rozeznat, kde přesně chyba nastala.

5 Implementace

V této kapitole je popsána implementace jednotlivých aplikací. Jedná se o aplikace bežící na herním serveru a o mobilní aplikaci.

5.1 Klientská aplikace

Jak bylo zmíněno výše, pro front-end bude využit framework Angular. Tento framework je využívaný pro tvorbu jednostránkových webových aplikací. Po vytvoření projektu pomocí Angular CLI (Command Line Interface) je nutné vytvořit jednotlivé komponenty. Tyto komponenty jsou hlavními prvky celé aplikace. Všechny komponenty a ostatní soubory je možné generovat pomocí Angluar CLI [24]. S frameworkem Angular bude také použit framework Bootstrap, který zjednodušuje vytváření vzhledu webových stránek a aplikací.

Komponent ve frameworku Angular se skládá ze čtyř souborů. Prvním z nich je soubor s HTML (Hypertext Markup Language) kódem, dále se jedná o soubor s CSS (Cascading Style Sheets) komponentu, soubor s TypeScriptem a soubor pro jednotkové testy. První dva z těchto souborů jsou použity převážně pro vytváření vzhledu komponentu. Pravděpodobně nejdůležitější je soubor s TypeScriptem komponentu, ve kterém probíhá programování jednotlivých funkcí [24].

I když je většina logiky implementována v souborech komponentu, je pro některé funkce dobré využít tzv. služeb. Služby je možné generovat pomocí Angular CLI. Jedná se o TypeScriptové soubory, ve kterých je možné vytvářet funkce, a tyto funkce následně volat z jednotlivých komponentů. Pro většinu komponentů byly vytvořeny služby pro volání API.

Komponenty v aplikaci jsou rozděleny do dvou kategorií, sdílené stránky a klasické stránky. Sdílené stránky jsou využity vždy v celé aplikaci. Jedná se o navigační menu na začátku stránky a o zápatí stránky. Navigační menu obsahuje odkazy na jednotlivé další komponenty webové aplikace. Změna jednotlivých komponentů je provedena pomocí *routeru*. *Router* ve framewroku Angular umožňuje přidat cesty k jednotlivým komponentům. Příklad vytvořeného *Routeru* je ve výpisu 5.1. Z výpisu kódu je zřejmé, že v *routeru* se vytváří vazba mezi cestou a daným komponentem. Položka *canActivate* slouží k omezení přístupu k cestám dle dané podmínky. Ochrana cest bude popsána později v této kapitole.

Výpis 5.1: Vytvoření cest v routeru.

```
1 const routes: Routes = [
2   {path: '', component:LoginComponent},
3   {path: 'domu',component:MainComponent, canActivate: [
       AuthGuardGuard]},
4   {path: 'sberatelske-predmety', component:
       CollectiblesComponent, canActivate: [AuthGuardGuard]},
```

Router je možné použít jak z HTML kódu, tak z TypeScript kódu. V případě navigačního menu je *router* použit z HTML kódu. V navigačním menu jsou odkazy na jednotlivé komponenty, a také menu, které obsahuje tlačítko pro přechod na stránku s informacemi o momentálně přihlášením účtu a pro odhlášení uživatele. Část kódu navigačního menu je ve výpisu 5.2. Komponent zápatí neobsahuje žádné funkce a slouží pouze ke zlepšení vzhledu stránky.

Výpis 5.2: Část navigačního menu a použití routeru.

```
1 <li class="nav-item">
2     <a class="nav-link" routerLink="domu" routerLinkActive="
         active-link">Hlavní strana</a>
3 </li>
4 <li class="nav-item">
5     <a class="nav-link" routerLink="sberatelske-predmety"
         routerLinkActive="active-link">Sběratelské předměty</a
         >
6 </li>
```

Pomocí Angular CLI bylo následně vygenerováno dvanáct komponentů, které jsou použity pro implementaci ostatních stránek. Jako první byly vygenerovány komponenty pro volně přístupné stránky. Jedná se o komponent *login* a *register*. Tyto komponenty obsahují kód pro stránku přihlášení a registraci uživatele. Oba tyto komponenty obsahují formulář, který uživatel vyplní buď pro registraci, nebo pro přihlášení. Komponent *login* obsahuje funkce pro inicializaci formuláře a přihlášení. Funkce *login* využívá pro přihlášení službu *authService*. Tato služba implementuje několik funkcí. První z nich je funkce *login*. Tato funkce slouží k přihlášení a získání tokenů z backendu. Funkce *logout* slouží k odhlášení uživatele. Z úložiště jsou odstraněny tokeny, je zrušeno naplánované obnovení tokenů a uživatel je pomocí *routeru* přesměrován na stránku s přihlášením. Funkce *setSession* slouží k uložení tokenů do úložiště a získání dat z tokenů. Uživatel je následně přesměrován na hlavní stranu aplikace. Funkce *startRefreshTokenTimeout* slouží k naplánování obnovení tokenů. Poslední funkce, *isLoggedIn*, slouží k získání informace, zda je uživatel momentálně přihlášen. Tato funkce kontroluje, zda je přístupový token platný.

Komponent *register* je velmi podobný předchozímu komponentu. Komponent obsahuje funkce pro inicializaci formuláře a funkci pro registraci. I pro tento komponent byla vygenerována služba. Jedná se o službu *apiService*. Tato služba je také použita pro implementací funkcí komunikace s API. Tuto službu tedy budou využívat i všechny ostatní komponenty. Komponent *register* využívá z této služby funkci pro registraci.

Ve službě *apiService* jsou implementovány všechny funkce pro volání API. První z těchto funkcí je funkce *register*. Tato funkce vytváří POST HTTP požadavek, kde tělem tohoto požadavku jsou uživatelské informace potřebné pro registraci. Po úspěšné registraci užvatele je uživatel přesměrován na stránku s přihlášením. Tato funkce je ve výpisu 5.3.

Výpis 5.3: HTTP požadavek na registraci uživatele.

```
1 register(player:any){
2     let url = "https://dpapp.onrender.com/register/"
3     const body = {username: player.name, email: player.email,
          password: player.password}
4     return this.http.post(url, body, {headers: this.
         httpHeaders}).subscribe(() => {this.router.navigate(['
         /'])}, err =>{console.log(err);})}
```

Další funkcí je *changeCache*. Tato funkce slouží úpravě informací o keši jejím vlastníkem. Funkce odešle v požadavku upravenou keš. Backend tento požadavek zpracuje, keš upraví a odešle upravenou keš zpět v odpovědi. Tato nová keš je poté vrácena do komponentu, ze kterého byla tato funkce volána. Funkce *getCurrentItem* slouží k získání putovního předmětu, který uživatel momentálně vlastní. *GetAccountInto* je funkce, která získává informace o momentálně přihlášeném hráči. Funkce vezme z úložiště ID hráče, které bylo v tokenu, a získá informace o hráči s tímto ID. Další funkcí je *getCaches*. Tato funkce získává informace o všech dostupných keších. Funkce *getPlayerCaches* získává informace o keších, které daný hráč vlastní. Dále je implementována funkce *getPlayerItems*, která slouží k získání putovních předmětů, které hráč vytvořil. Pro získání informací o keších, kterými putovní předmět prošel, byla implementována funkce *getCardsFromItem*. Jelikož jsou v databázi souřadnice keše ukládány v jiné podobě, než jakou je možné použít v mapě, bylo nutné vytvořit funkci, která získá správný formát těchto dat. Funkce pro převod do vhodného formátu byla implementována v back-endu, je však nutné tyto data přenést na front-end. Pro získání těchto dat byla implementována funkce *getMarkers*. Poslední dvě funkce v této službě jsou *getClickedCard* a *saveItem*. První z těchto funkcí slouží k získání informací o jedné dané keši. Druhá z těchto funkcí, *saveItem*, slouží k vytvoření nového putovního předmětu.

Následně byly vytvořeny komponenty, které pro přístup vyžadují přihlášeného uživatele. Hlavním komponentem je *main*. V tomto komponentu je implementována mapa, na které jsou zobrazeny všechny existující keše. Pro zobrazení všech keší je použit cyklus *ngFor*, který umožňuje dynamicky přidávat HTML prvky [24]. Část kódu, která je použita pro vytvoření značek keší, je ve výpisu 5.4.

Výpis 5.4: Vytvoření několika značek pomocí ngFor.

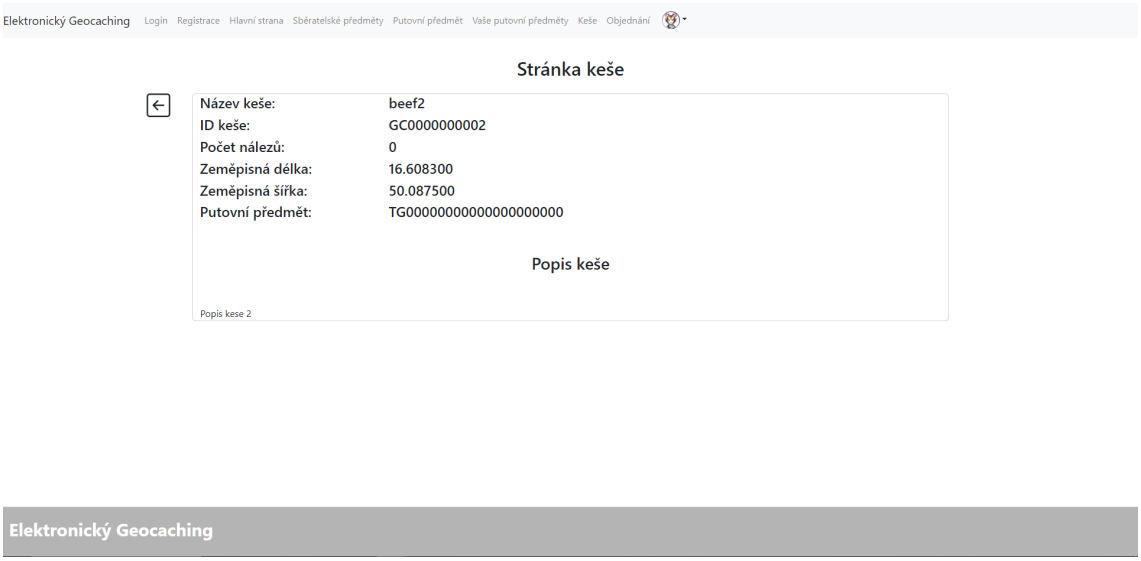
```
1 <google-map height="700px" width="100%" [center]="center" [
     zoom]="7">
2         <map-marker *ngFor="let␣cache␣of␣caches"
3         [position]="cache.position"
4         [label]="cache.nazev.toString()"
5         (mapClick)="cacheInfo(cache.id)"
6         (mapClick)="showInfo()">
7         </map-marker>
8 </google-map>
```

Po kliknutí na některou ze značek jsou zobrazeny některá informace o dané keši. U informací o keši je také tlačítko, které umožňuje uživateli přejít na stránku dané keše, která obsahuje více informací o dané keši. Komponent také zobrazuje některé základní informace o uživateli. Komponent také využívá dvou služeb pro správnou funkčnost. První z nich je již zmíněná služba *apiService*. Tato služba je využita pro získání informací o hráči a také pro získání informací všech keší. Druhá služba je *GetcarddetailService*. Tato služba slouží pouze pro dočasné uchování vybrané keše, nebo předmětu. Funkce *setCache* uloží danou keš pomocí zmíněné služby a přesměruje uživatele na komponent *cacheinfo*.

Tento komponent slouží k zobrazení informací o jedné keši. Data této keše jsou získána pomocí služby *GetcarddetailService*. Tento komponent také obsahuje tlačítko, které uživatele přesměruje zpět na původní komponent 5.1.

Komponent *collectibles* je určen pro zobrazení sběratelských předmětů hráče. Sběratelské předměty jsou obrázky, které uživatel postupně získává. Pokud není sběratelský předmět získán, je obrázek rozmazán. Komponent také ukazuje počet sběratelských předmětů, které hráč získal, a které je možné celkově získat. Komponent používá službu *apiService* pro získání dat o hráči. Z těchto dat jsou vyjmuty data o sběratelských předmětech hráče.

Dalším komponentem je *item*. Tento komponent zobrazuje putovní předmět, který hráč momentálně vlastní. S využitím služby *apiService* získá komponent putovní předmět hráče. Pokud je putovní předmět prázdný, zobrazí se obrázek prázdného putovního předmětu. Pokud putovní předmět existuje, zobrazí se obrázek běžného putovního předmětu.

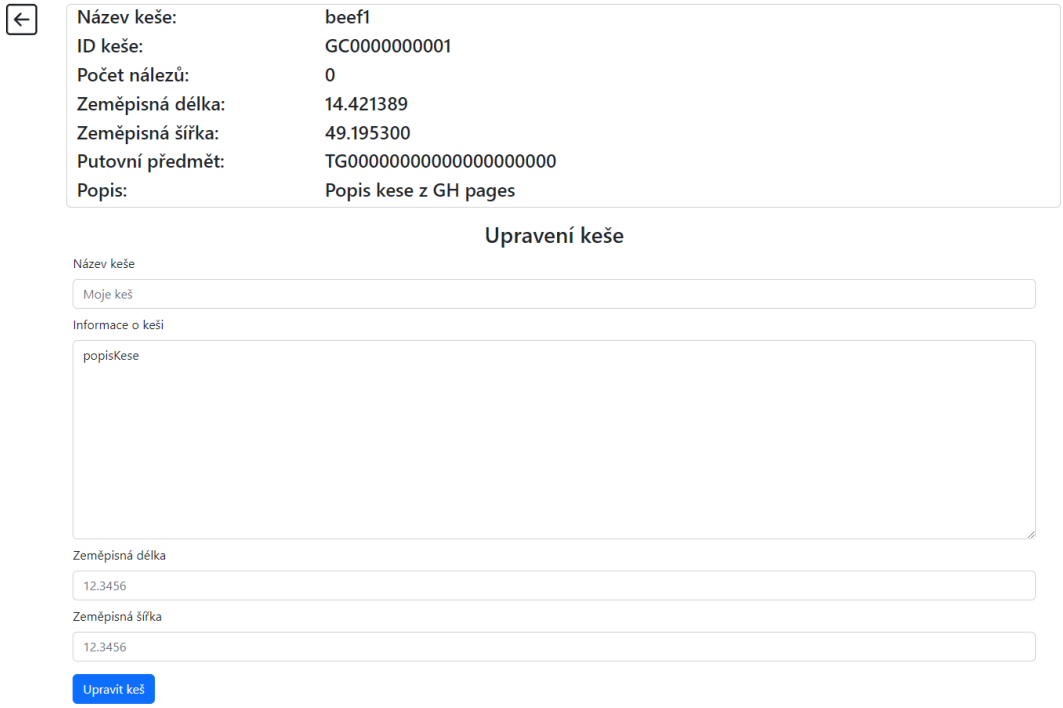


Obr. 5.1: Komponent s informacemi o keši.

Traveling je komponent, který zobrazuje všechny putovní předměty, které uživatel vytvořil. Zde je také umožněno uživateli vytvořit nový putovní předmět, pokud momentálně žádný nevlastní. Tento komponent používá dvě služby. První z nich je služba *apiService*, díky které komponent získá informace o putovních předmětech hráče a je schopen vytvořit nový putovní předmět. Druhou službou je *GetcarddetailService*, která slouží pro dočasné uložení putovního předmětu. Po kliknutí na putovní předmět je uživatel přesměrován na komponent, který obsahuje informace o daném putovním předmětu.

Pro zobrazení informací o daném putovním předmětu byl vytvořen komponent *itemdetail*. Tento komponent zobrazuje obrázek putovního předmětu a jeho základní informace. Pokud putovní předmět prošel nějakou keší, nebo se v jedné nachází, je také zobrazena cesta tohoto putovního předmětu. Základním zobrazením této cesty je seznam keší, ve kterých se předmět nacházel. Komponent také obsahuje tlačítko, které zobrazí mapu. Na této mapě jsou umístěny značky keší, kterými předmět prošel. Po kliknutí na značku jsou zobrazeny informace o keši a je možné přejít na stánku s informacemi o keši.

Komponent *caches* je určen pro zobrazení keší, které uživatel vlastní. I tento komponent využívá, krom *apiService*, také službu *GetcarddetailService*, díky které je možné dočasně uložit jednu keš. Po kliknutí na danou keš jsou její informace uloženy a uživatel přesměrován na komponent, který je určen pro zobrazení a úpravu informací keše. Tímto kompoenntem je *cachedetail*. Tento komponent je velmi podobný komponentu *cacheinfo*, ale krom zobrazení informací obsahuje také formulář, díky kterému je možné tyto informace měnit 5.2.



Obr. 5.2: Komponent možností úpravy informací keše.

Posledními dvěma komponenty jsou *order* a *account*. První z nich, *order*, je určen pro vytvoření objednávky keše. Tento komponent obsahuje formulář, do kterého uživatel vyplní potřebné informace. Po kliknutí na tlačítko je objednávka odeslána na back-end, kde je uložena. Komponent *account* je určen pro zobrazení informací o uživateli.

Vytvořená aplikace byla nahrána na platformu github a zpřístupněna pomocí platformy github pages. Aplikace je přístupná z [25].

5.2 Serverová aplikace

Back-end herního serveru byl vytvořen ve frameworku Django. Po vytvoření aplikace je nejprve nutné upravit nastavení aplikace. Nastavení se nachází v souboru *settings.py*. Zde je nutné přidat do sekce *INSTALLED_APPS* vytvořenou aplikaci a ostatní nainstalované balíky, které jsou použity. Dále je nutné upravit záznam o databázi. Je třeba zadat správný název databáze a přihlašovací údaje [26]. V tomto souboru také probíhá nastavení JWT. V aplikaci bylo použito základní nastavení, u kterého byla změněna doba platnosti.

V souboru *modles.py* byly implementovány vytvořené modely. Tyto modely se liší od původně navržených. Důvodem změn jsou problémy s implementací navržených

modelů a také přidání některých dalších funkcí. V tomto souboru byly vytvořeny modely hráče, karty (keše) a putovního předmětu.

Vzhledem k funkci vazby mezi sběratelským předmětem a hráčem byl model pro sběratelský předmět odstraněn a místo něj bylo přidáno další pole k modelu hráče. Model hráče nově obsahuje pole typu *JSONFiled*, do kterého jsou uloženy informace o sběratelských předmětech hráče. Další změny u modelu hráče je vazba model *User*. Tento model je jedním ze základních modelů framewroku Django. Díky tomuto modelu je možné vytvářet a autentizovat uživatele dle jejich přihlašovacích údajů. Dále bylo k modelu hráče přidáno pole s počtem otevřených keší a také pole s datem registrace uživatele. Model také obsahuje funkce pro vytvoření základních hodnot pro nepovinná pole. Jedná se v podstatě o formu inicializace polí, aby bylo možné jej uložit do databáze. Dalším modelem je model pro putovní předmět. K tomuto modelu bylo přidáno pole typu *ImageFiled*, které umožňuje nahrání obrázku na server. I u tohoto modelu je vytvořena funkce pro vytvoření některých základních hodnot putovního předmětu. Posledním implementovaným modelem je model karty. Samotný model karty je v podstatě stejný jako v návrhu s jedinou výjimkou. K tomuto modelu bylo přidáno pole s popisem keše, kam může vlastník této keše vložit jeho vlastní text. Model dále obsahuje dvě funkce pro vytvoření základních informací o kartě. Jedná se o vytvoření ID keše a putovního předmětu v keši. Tyto dvě funkce vrací předem stanovený řetězec znaků. Příklad tohoto modelu je ve výpisu 5.5.

Výpis 5.5: Model hráče.

```
1  class Karta(models.Model):
2      def card_id_default():
3          return "GC0000000000"
4      def putovni_predmet_default():
5          return "TG00000000000000000000"
6      idkarty = models.CharField(max_length=12)
7      nazev = models.CharField(max_length=30, default='beef')
8      vlastnik = models.CharField(max_length=30)
9      zemdelka = models.DecimalField(max_digits=9,
           decimal_places=6)
10     zemsirka = models.DecimalField(max_digits=9,
           decimal_places=6)
11     putpredmet = models.CharField(max_length=30, default=
           putovni_predmet_default)
12     pocetnalezu = models.IntegerField(default=0)
13     popis = models.TextField(default='')
```

Po vytvoření modelů je možné provést migraci dat do databáze. Migrace dat do databáze se provádí pomocí dvou příkazů. Prvním z nich je příkaz *manage.py make-*

migrations. Tento soubor vytváří migraci, která je následně aplikována na databázi. Pro provedení této migrace je použit příkaz *manage.py migrate* [27].

V aplikaci byly dále vytvořeny tzv. serializátory (anglicky serializer). Serializátor slouží k převedení dat získaných z databáze na datové tyopy, které je možné jednoduše použít v Pythonu [26]. V serializátoru je definován model, se kterým serializátor pracuje, a také pole modelu, která má zpracovat a převést. Příklad serializátoru hráče je ve výpisu 5.6. Serializátory byly implementovány v souboru *serializers.py*

Výpis 5.6: Serializátor hráče.

```
1 class HracSerializer(serializers.ModelSerializer):
2     class Meta:
3         model = Hrac
4         fields = ['id','username','user','mail','otevrenekese
              ','otevrenekesepoc','putpredmet','putpredmety','
              sberpredmet','pocetsberpred','vlastnenekese','
              registrace','player_id']
```

Nejdůležitější částí této aplikace jsou tzv. *views*. Jedná se o třídy, které zpracovávají HTTP požadavky a vrací odpovědi [18]. *Views* byly implementovány v souboru *views.py*. Vytvořené *views* dědí ze tříd z Django REST frameworku. Jedná se o třídy *ModelViewSet* a *APIView*.

Třídy všech modelů dědí ze třídy *ModelViewSet*. Jako první byla implementována třída hráče. V této třídě je definována funkce *retrieve*. Tato funkce bere ID, které je v příchozím HTTP požadavku, a podle toho ID vyhledá hráče v databázi. Následně tento výsledek převede do formátu JSON (JavaScript Object Notation) a odešle odpověď obsahující tato data. Vytvořený *view* pro model hráče je ve výpisu 5.7.

Výpis 5.7: View hráče.

```
1 class hrac_viewset(viewsets.ModelViewSet):
2     queryset = Hrac.objects.all()
3     serializer_class = HracSerializer
4     def retrieve(self, request, *args, **kwargs):
5         params = kwargs
6         hrac = Hrac.objects.get(id= params['pk'])
7         serializer = HracSerializer(hrac)
8         return Response(serializer.data)
```

Jako další byla implementována třída pro kartu (keš). Tato třída, krom funkce *retrieve*, implementuje také funkci *list*. Tato funkce vrací všechny objekty typu karta v databázi. Funkce *retrieve* je podobná jako u třídy hráče. Pomocí ID z HTTP požadavku vybere karty, které hráč vlastní. Data o všech těchto kartách jsou poté odeslána v odpovědi. Jako poslední byla implementována třída pro model putovního

předmětu. Zde je definována funkce *retrieve*. Tato funkce je, podobně jako u předchozích, určena k získání jednoho nebo více sběratelských předmětů hráče. Jedná se o předměty, které hráč vytvořil.

Dále byly vytvořeny třídy, které dědí z *APIView*. Jako první byla vytvořena třída *put_predmet_pozice*, která má definovánu funkci *get*. Tato funkce vrací putovní předmět, který se nachází na pozici definované v HTTP požadavku. Třída *ceta_predmetu* je použita k získání cesty daného putovního předmětu. Ve funkci *post* je z HTTP požadavku získáno pole, které obsahuje ID karet, kterými putovní předmět prošel. Tyto karty jsou vyhledány v databázi a z jejich informací je vyjmuta pozice a ID. Tyto dvě informace jsou poté odeslány v odpovědi. V souboru byly dále implementovány třídy *card_info* a *get_all_cards*. První z těchto tříd je určena pro získání informací o několika kartách. Druhá je určena pro získání informací všech karet a vrácení pouze vybraných dat z těchto informací v odpovědi. Následně je v souboru implementována třída *card_detail*. Tato třída umožňuje získání informací o jedné dané kartě a také implementuje funkci *put*. Tato funkce umožňuje úpravu informací o kartě v databázi. V této funkci je nejdříve získán daný objekt karty z databáze. Data této karty jsou poté upravena dle dat přidaných k HTTP požadavku a karta je uložena. Třída *order_card* je určena ke zpracování objednávek karet. Funkce *post* vytváří nový textový dokument, do kterého jsou uloženy informace o objednávce karty. Vytváření souborů objednávek je nastaveno tak, že soubor má název ve tvaru *<příjmení>_<jméno>.txt*. Kartu vytváří administrátor na administrátorských stránkách serveru. Následně byla implementována třída *user_register*, která slouží k registraci nových uživatelů. Aplikace nejdříve vytvoří nového uživatele dle modelu *User*. Po vytvoření uživatele je vytvořen nový hráč, který má na tohoto uživatele vazbu. Po vytvoření hráče jsou vytvořeny jeho klíče a certifikát podepsaný certifikační autoritou. Další implementovanou třídou je *item_upload*. V této třídě je definována funkce *post*, ve které je vytvořen nový putovní předmět. Před samotným vytvořením předmětu je zkontrolováno, zda již u hráče není jiný putovní předmět. Pokud ne, je předmět vytvořen. Následující implementovanou třídou je *open_cache*. Funkce *post* v této třídě získá hráče, který požadavek odeslal, a keš, kterou hráč otevřel. Tyto informace obsahuje HTTP požadavek, který odeslala android aplikace. Po získání hráče a keše jsou v databázi nalezeny jejich putovní předměty. Provede se přepsání předmětů hráče a keše v databázi a u hráče proběhne získání nového sběratelského předmětu. U putovních předmětů je přepsáno ID jejich pozice a přenastavena hodnota pozice. Do pole putovního předmětu, které zaznamenává karty, kterými předmět prošel, je poté uloženo ID karty. Jednou z posledních implementovaných tříd je *get_cert*, která načte soubor certifikátu hráče a vrátí jej v odpovědi.

Posledními třídami implementovanými v souboru *view.py* jsou třídy pro JWT. Zde je k tokenu přidán záznam o uživatelském jménu hráče. Uživatelské jméno je

poté přidáno do těla tokenu a je možné jej po dekódování použít. Aplikace také obsahuje dva vytvořené Python soubory *idhelper.py* a *certhelper.py*. Soubor *idhelper.py* obsahuje funkce pro tvorbu ID hráče, keše a putovního předmětu. Druhý soubor, *certhelper.py*, obsahuje funkci pro vytvoření klíčů a certifikátu hráče a karty pomocí openssl. Tato funkce používá již vytvořené klíče certifikační autority. Tyto soubory byly také vytvořeny pomocí programu openssl. Byl také vytvořen certifikát certifikační autority.

Vytvořené *views* je nutné zpřístupnit na serveru. Soubor *urls.py* obsahuje jednotlivé cesty vytvořeného API. Zde jsou pomocí funkce *path* definovány cesty a *views*, které k těmto cestám patří [18]. Po přidání potřebných cest je možné získat data z databáze.

Framework Django obsahuje stránky pro správu aplikace. Tyto stránky budou použity pro vytváření nových karet. Pro přístup k těmto stránkám musí mít uživatel potřebná oprávnění. Díky tomu nebude možné, aby si kterýkoliv uživatel vytvořil vlastní keš. Aby bylo možné keš nebo jiný model vytvořit, je nutné jej registrovat. Registrace modelu se provádí v souboru *admin.py* [26].

Serverová aplikace byla nahrána na platformu render. Na této platformě byla také vytvořena příslušná databáze. Aplikace je přístupná z [28]. Vzhledem k omezení na této platformě byly některé funkce vypnuty. Aplikace nevytváří a nevrací certifikáty, jelikož platforma nemá nainstalováno openssl.

5.3 Android aplikace

V android aplikaci byly vytvořeny tři aktivity a čtyři fragmenty. Ke všem těmto souborům je automaticky vytvořen jejich příslušný soubor s rozložením a vzhledem. První z nich je *MainActivity*, která slouží pro přihlášení uživatele. Po stisknutí tlačítka pro přihlášení je volána funkce *post*. Tato funkce vytváří HTTP POST požadavek, který je odeslán na back-end. V odpovědi se nachází tokeny pro přístup a obnovení, které jsou v aplikaci uloženy pomocí funkce *saveTokens*. Z JWT v odpovědi je také získáno uživatelské jméno a ID uživatele. Po úspěšném přihlášení je spuštěna druhá aktivita, *MainPage*.

Aktivita *MainPage* obsahuje kontejner, ve kterém je možné zobrazit jednotlivé fragmenty. Dále je v aktivitě implementováno menu na spodní straně obrazovky, které obsahuje tlačítka pro zobrazení jednotlivých fragmentů. Uprostřed menu je implementováno plovoucí tlačítko, které uživateli zobrazí mapu s polohou keší a uživatele. Spodní menu je definováno v souboru *bottom_nav.xml*, kde byly vytvořeny čtyři viditelné položky a jedna skrytá. Skrytá položka je vytvořena kvůli plovoucímu tlačítku, které bez této položky zasahuje do ostatních možností menu, viz obrázek 5.3.

Obr. 5.3: Spodní menu s a bez skryté položky.

Po stisknutí některé z možností spodního menu je v kontejneru zobrazen příslušný fragment. Pro zobrazení fragmentu mapy je nutné stisknout vytvořené plovoucí tlačítko. V aktivitě je také vytvořeno nové vlákno. Toto vlákno je použito v metodě *refreshTokens*, která po uplynutí doby tokenu tento token obnoví pomocí HTTP požadavku. Dále je v aktivitě implementována metoda *loadData*, která načte uložená data uživatele z úložiště.

Jak již bylo zmíněno výše, aplikace obsahuje čtyři fragmenty, které je možné zobrazit v aktivitě *MainPageActivity*. Jedná se o fragment hlavní strany (*MainPageFragment*), fragment sběratelských předmětů (*CollectiblesFragment*), fragment mapy (*MapsFragment*) a fragment putovního předmětu (*ItemFragment*). Fragment hlavní strany je základní fragment, který je načten ihned po spuštění *MainPageActivity*. Tento fragment zobrazuje základní informace o uživateli. Ve fragmentu jsou implementovány dvě metody. První z nich, *loadData*, slouží, stejně jako u aktivity, k načtení uložených dat. Druhá metoda *get* slouží k získání dat hráče z herního serveru. Fragment sběratelských předmětů implementuje stejné metody a navíc také metodu *setItemImage*, která dle stavu nalezení sběratelského předmětu buď zobrazí daný předmět nebo obrázek zvolený pro nenalezený předmět. Fragment mapy slouží k zobrazení polohy hráče a všech keší na mapě. Pro přístup k poloze hráče je nutné získat daná povolení od uživatele k použití jeho polohy. Tento fragment také implementuje metodu pro načtení uložených dat a metodu *get*, ve které jsou získány polohy všech keší. Z odpovědi na HTTP požadavek je vytvořeno JSON pole. Pomocí cyklu *for* jsou z jednotlivých položek tohoto pole vytvořeny značky keší na mapě, viz výpis 5.8.

Výpis 5.8: Vytvoření značek keší na mapě.

```
1 JSONArray jarray = new JSONArray(responseString);
2 for (int i = 0; i< jarray.length(); i++){
3     JSONObject json = jarray.getJSONObject(i);
4     JSONObject position = json.getJSONObject("position");
5     LatLng marker = new LatLng(parseDouble(position.getString
          ("lat")), parseDouble(position.getString("lng")));
6     map.addMarker(new MarkerOptions().position(marker).title(
          json.getString("nazev")));
7 }
```

Dále byl vytvořen fragment putovního předmětu. Fragment putovního předmětu, podobně jako předchozí, implementuje metody pro načtení uložených dat a získání dat z herního serveru. Po získání dat z herního serveru je zobrazen putovní předmět hráče. Pokud hráč žádný putovní předmět nevlastní, je ID toho předmětu nastaveno na ID prázdného předmětu a stejně tak i obrázek tohoto putovního předmětu.

Jako poslední byla implementována aktivita *NfcActivity*. V této aktivitě se nachází převzatý kód pro komunikaci s kartou. Aktivita také implementuje několik nových funkcí. Funkce *loadData* slouží, podobně jako u předchozích aktivit, k načtení uložených dat. Další dvě nové funkce jsou *get* a *post*. Funkce *get* slouží k získání dat o hráči z herního serveru. Z těchto informací je vyjmuto pole, které obsahuje otevřené keše. Toto pole je poté použito v převzatém kódu pro kontrolu otevřených keší. Funkce *post* předává hernímu serveru informace o hráči a otevřené keši. Herní server tyto data zpracuje a provede potřebné změny v databázi, aby stav putovních předmětů v databázi odpovídal reálnému stavu putovních předmětů na kartě a v aplikaci. Další novou funkcí je *openCache*, která přidá do pole otevřených keší nově otevřenou keš, a volá již zmíněnou funkci *post*, viz výpis 5.9. Tato funkce následně zobrazí uživateli hlášku, že otevřel danou keš. Aktivita také nově obsahuje dvě funkce, které zobrazí menu v pravé horní části obrazovky, a přepnou hráče zpět na *MainPage* aktivitu.

Výpis 5.9: Funkce otevření keše v aplikaci.

```
1 private void openCache() {
2         IDs.add(new String(S));
3         post(user_id,new String(S));
4         Toast.makeText(this,"Úspěšně␣jste␣otevřeli␣keš:␣"+new
              String(S), Toast.LENGTH_LONG).show();
5     }
```

Vzhledem k novým funkcím bylo nutné provést úpravy původního kódu. Do původní funkce, která zajišťovala běh protokolu, bylo přidáno volání metody *openCache*. Metoda je volána po úspěšném dokončení protokolu. V původní funkci zajišťující kontrolou otevřených keší (*checkCacheList*) byla provedena úprava. Původní kód hledal pole bajtů v poli textových řetězců. Funkce byla upravena, aby se pole bajtů převedlo na textový řetězec. Upravený kód je ve výpisu 5.10.

Výpis 5.10: Kontrola otevřených keší.

```
1 private boolean checkCacheList(byte[] id) throws IOException
2     {
3         boolean result = IDs.contains(new String(id));
4         return result;
5     }
```

Původní funkce pro získání a zobrazení sběratelských předmětů již není použita. Odemknutí sběratelského předmětu probíhá na herním serveru. Pro zobrazení sběratelských předmětů byl vytvořen již dříve zmíněný fragment. Tyto funkce tedy nejsou použity.

5.4 Applet karty

Applet karty byl převzat v plném rozsahu. V kódu však byla opravena chyba, která znemožňovala znovu použití dané karty. V původním kódu je použita proměnná *stop*, která určuje, zda karta přijala příkaz k ukončení běhu protokolu. Na začátku běhu protokolu je tato proměnná zkontrolována a dle ní karta buď spustí protokol, nebo odešle chybovou hlášku (karta předpokládá že protokol stále běží, a proto nespustí další běh protokolu). Do kódu byla přidána funkce *deselect*, která nastaví hodnotu proměnné *stop* na původní hodnotu, jakmile karta přijme příkaz k zastavení protokolu.

5.5 Vytváření karet

Pro vytváření karet byly použity dva již existující programy. Pro vytváření potřebných souborů byl použit *ant-javacard*. Byl vytvořen potřebný soubor *build.xml*, dle kterého je vytvořen CAP soubor. Samotné změny kódu (například jiné ID karty) jsou provedeny ručně, přímo v kódu appletu. Pro instalaci appletu na kartu byl použit *GlobalPlatformPro*. Pro oba tyto programy je nutné, aby zařízení mělo nainstalováno Javu. Oba tyto programy jsou kompatibilní jak s operačním systémem Windows, tak i s Linuxem.

5.6 Certifkáty

Součástí celé infrastruktury je také certifikační autorita, která vydává certifikáty hráčům a keším. Tato certifikační autorita je vytvořena na herním serveru. Součástí serverové aplikace jsou i funkce k vytvoření jednotlivých certifikátů. Bohužel po otestování vytvořených certifikátů bylo zjištěno, že původní certifikáty v mobilní aplikaci a appletu karty jsou nestandardní. Obě aplikace jsou vytvořeny takovým způsobem, že fungují pouze s původními certifikáty. Obě implementace vybírají data z certifikátu velmi specifickým způsobem a při použití certifikátu, který byl vytvořen herním serverem, jsou data chybná. Přepsání implementace práce s certifikáty je bohužel nad rámec této práce. Příklad získání ID hráče z certifikátu je ve výpisu 5.11.

Výpis 5.11: Získání ID z certifikátu v appletu karty.

```
1  private void getIDFromCert(byte[] CRT, byte[] id) {
2          short avpCertHeaderLen = (short) 3;
3          short dataOffs = (short) (avpCertHeaderLen + 4);
4          short dataLen = (short) (Util.getShort(CRT, (short)(
               dataOffs + 2)) + 4);
5          short offs1 = (short) (dataOffs + 4);
6          short len1 = (short) (CRT[(short)(offs1 + 1)] + 2);
7          short offs2 = (short) (offs1 + len1);
8          short len2 = (short) (CRT[(short)(offs2 + 1)] + 2);
9          short offs3 = (short) (offs2 + len2);
10         short len3 = (short) (CRT[(short)(offs3 + 1)] + 2);
11         short offs4 = (short) (offs3 + len3);
12         short len4 = (short) (CRT[(short)(offs4 + 1)] + 2);
13         short offs5 = (short) (offs4 + len4);
14         short offs6 = (short) (offs5 + 2);
15         short offs7 = (short) (offs6 + 2);
16         short offs8 = (short) (offs7 + 2);
17         short len8 = (short) (CRT[(short)(offs8 + 1)] + 2);
18         short offs9 = (short) (offs8 + len8);
19         short len9 = (short) (CRT[(short)(offs9 + 1)] + 2);
20         short offsH = (short) (offs9 + 2);
21         short lenH = (short) (len9 - 2);
22         Util.arrayCopyNonAtomic(CRT, (short)(offsH), id, (
               short)0, (short)lenH);
23     }
```

Ve výpisu 5.12 je výstup s použitím původního certifikátu a certifikátu vytvořeného herním serverem.

Výpis 5.12: Výsledek funkce s původním a novým certifikátem.

```
1 Vystup původního
2 P998765432100123456789
3 Výstup nového
4 567890F0*   H   =+? 2       ClnJ6j       |~  /       g
```

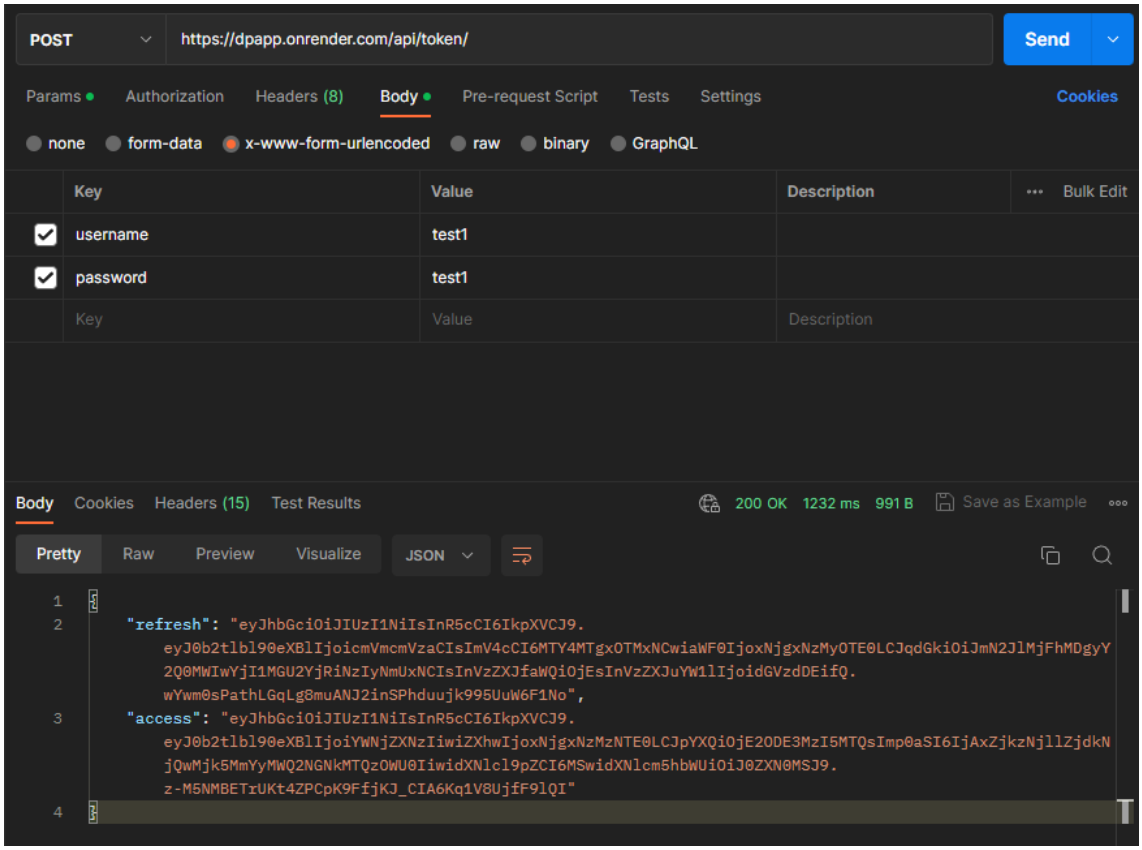
Dalším problémem při práci s certifikáty jsou samotné karty. Pro implementaci byly použity karty Java Card verze 2.2.2, které nepodporují práci s certifikáty X.509 [29]. Z tohoto důvodu applet nepracuje s objektem certifikátu, ale s bajtovým polem. Pro implementaci je vhodnější karta, která podporuje práci s certifikáty X.509. Příkladem může být Java Card verze 3.1, která certifikáty X.509 podporuje, [30]. Z těchto důvodů budou v aplikacích použity původní certifikáty.

6 Testování

Tato kapitola obsahuje testování jednotlivých částí infrastruktury. Bylo provedeno testování herního serveru i Android aplikace. Pro účely testování byli vytvořeni dva uživatelé, kteří mají uživatelská jména test1 a test2, a hesla, která se shodují s jejich uživatelskými jmény. Uživatel test1 je také administrátorem a může se přihlásit na administrátorské stránky serverové aplikace. Pomocí těchto stránek byly vytvořeny dvě keše, ke kterým byly vytvořeny i příslušné bezkontaktní čipové karty.

6.1 Serverová aplikace

Pro testování backendu byl použit program Postman. Tento program umožňuje odesílat HTTP požadavky na danou adresu a zobrazit odpovědi serveru. Jako první bylo otestováno přihlášení a získání nových tokenů. V požadavku bylo uvedeno uživatelské jméno a heslo. Nejdříve bylo uvedeno správné uživatelské jménu a heslo. Z herního serveru byly získány příslušné tokeny, viz obrázek 6.1.



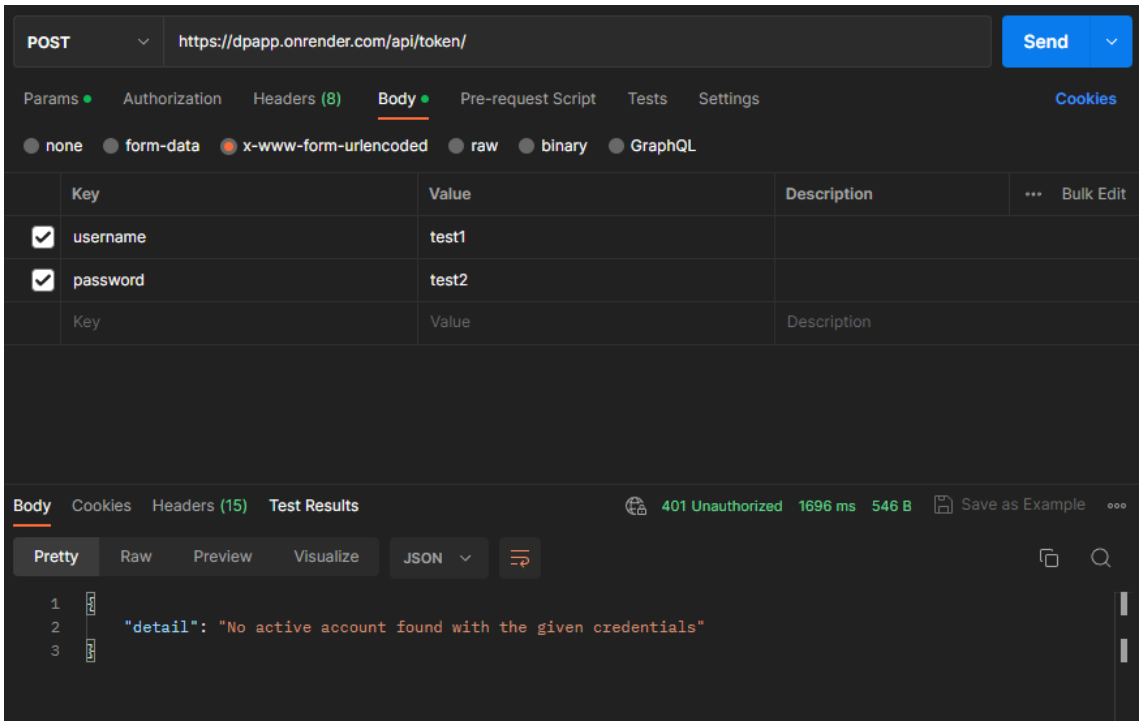
Obr. 6.1: Přihlášení se správnými údaji.

Získané tokeny je možné dekódovat a získat informace, které token obsahuje. Údaje z dekódovaného tokenu jsou ve výpisu 6.1. Z tokenu je možné získat ID hráče, jeho přezdívku a čas platnosti tokenu. V tokenu je také informace o typu daného tokenu, tedy zda se jedná o přístupový token (access token), nebo token pro obnovení tokenů (refresh token).

Výpis 6.1: Údaje z dekódovaného tokenu.

```
{
  "token_type": "access",
  "exp": 1681734066,
  "iat": 1681733466,
  "jti": "5062160d8a854086a4a2d63cfd75ff3d",
  "user_id": 1,
  "username": "test1"
}
```

Ve druhém testu bylo změněno heslo uživatele. V přijaté odpovědi se nachází informace o chybě. V tomto případě je zde informace, že se zadané údaje neshodují s žádnou položkou v databázi. Databáze tedy neobsahuje žádného uživatele, který má dané uživatelské jméno a heslo. Výsledek testu je na obrázku 6.2.



Obr. 6.2: Přihlášení s chybnými údaji.

Dalším cílem testování bylo získání dat z databáze. Pro přístup k datům je nutné použít platný přístupový token. Byly provedeny dva testy, kdy byly použity dva rozdílné tokeny. V prvním testu byl použit platný přístupový token pro získání dat z databáze. Tento test byl úspěšný a herní server odeslal příslušná data. Ve druhém testu byl použit chybný přístupový token. K přidání chyby to tokenu byl token mírně upraven, ale výsledek by byl totožný i při použití tokenu, kterému vypršel čas jeho platnosti, nebo při použití špatného typu tokenu (použití obnovovacího tokenu). Výsledek obou testů je na obrázku 6.3. Na obrázku vlevo je situace, kdy byl použit platný token. Serverová aplikace token přijala, a vrátila data o hráči. Na obrázku vpravo je situace, kdy byl použit neplatný token. Aplikace token nepřijala, a zobrazila chybu.

Obr. 6.3: Vlevo použití platného tokenu, vpravo použití neplatného tokenu.

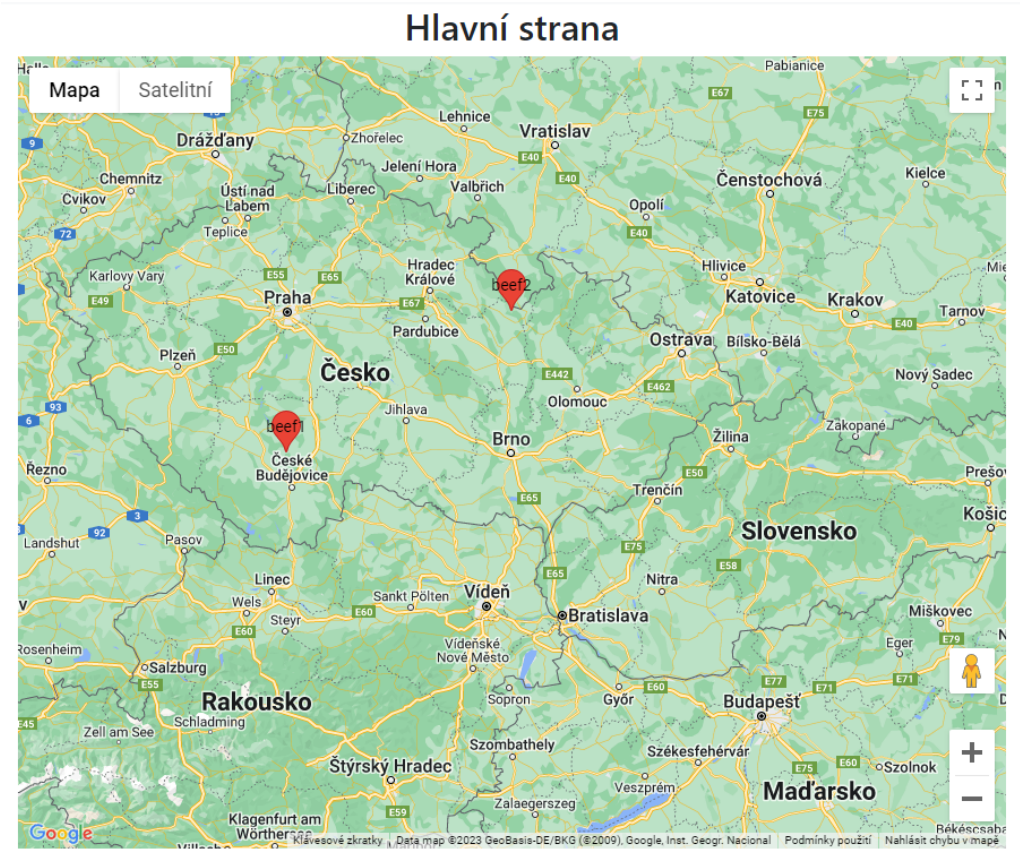
6.2 Klientská aplikace

Jako první bylo otestováno ukládání přijatých dat po přihlášení. Po otevření stránky bylo provedeno přihlášení. Po přihlášení a přesměrování na hlavní stránku bylo zkontrolováno úložiště v prohlížeči. V úložišti je možné vidět příslušné tokeny a také uživatelské jméno a ID, viz obrázek 6.4.

Key	Value
user_id	1
refresh_token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0b2tlbl90e...
username	test1
access_token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0b2tlbl90e...

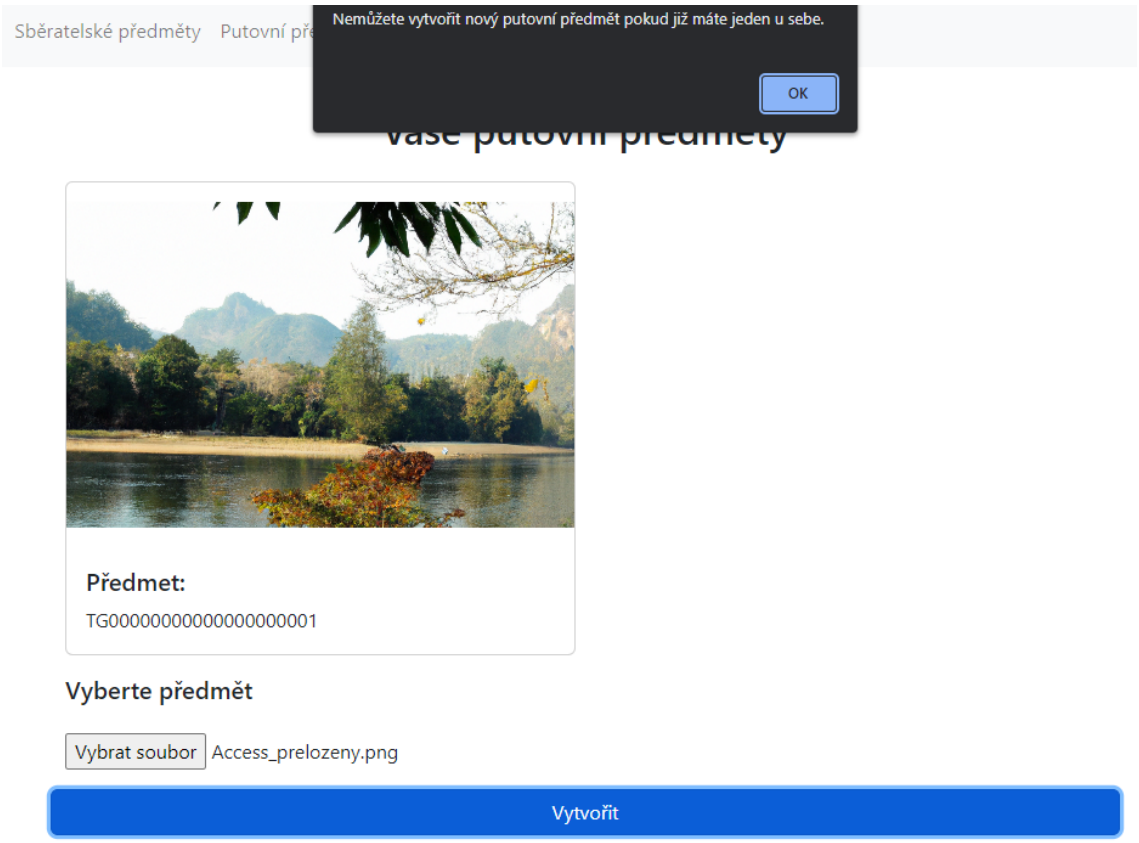
Obr. 6.4: Data uložená v prohlížeči.

Zároveň bylo na hlavní straně zkontrolováno, zda jsou keše viditelně na mapě. Obrázek 6.5 obsahuje mapu na hlavní straně klientské aplikace. Z obrázku je zřejmé, že obě vytvořené keše jsou na mapě.



Obr. 6.5: Mapa na hlavní straně klientské aplikace.

Dále byl proveden test putovních předmětů. Byl proveden pokus o vytvoření putovního předmětu. Přihlášený uživatel již má jeden putovní předmět u sebe, tato akce by tedy měla selhat. Po vytvoření předmětu bylo zobrazeno okno, které obsahovalo hlášku, že přihlášený uživatel již putovní předmět vlastní a nemůže tedy vytvořit nový putovní předmět. Výsledek tohoto testu je na obrázku 6.6.



Obr. 6.6: Pokus o vytvoření putovního předmětu.

Jako poslední byl proveden test objednání karty. Byl vyplněn formulář pro objednání a objednávka byla odeslána. Po odeslání objednávky byla v souborovém systému serverové aplikace vytvořena složka *objednavky* a v ní textový soubor nové objednávky. Tento soubor obsahoval data, která byla zadána ve formuláři. Obsah souboru je ve výpisu 6.2. Výpis byl upraven, aby jednotlivé položky byly na jednotlivých řádcích. V původním souboru byly všechny informace na jednom řádku.

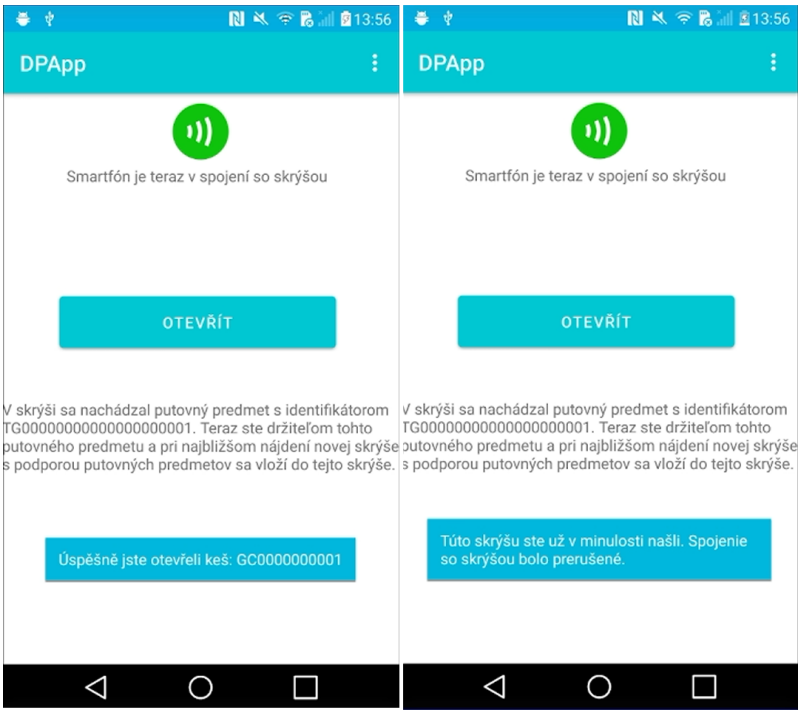
Výpis 6.2: Výpis ze souboru s objednávkou karty.

```
{
  "firstname": "jmeno",
  "lastname": "prijmeni",
  "city": "brno",
  "street": "Ul, 1",
  "psc": "12345",
  "phone": "123 456 789",
  "email": "test@test.com",
  "cachename": "Moje",
  "lng": "12",
  "lat": "34",
  "username": "test"
}
```

6.3 Android aplikace

Testování mobilní aplikace proběhlo na dvou zařízeních. První z nich je mobilní telefon LG G3 s Androidem verze 6.0. Tento mobilní telefon také disponuje rozhraním NFC. Druhým zařízením byl mobilní telefon Xiaomi Redmi Note 8 s Androidem verze 11.

V mobilní aplikaci byly otestovány některé základní úkony. Testování bylo zaměřeno na funkčnost komunikace s herním serverem a také na funkčnost opraveného kódu pro komunikaci se skrýší. První test bylo samotné otevření vytvořené keše. Po přiložení chytrého telefonu ke skrýši byl proveden pokus o otevření keše. Tento pokus byl úspěšný a na obrazovce se zobrazila informace o putovních předmětech a také o otevřené skrýši. Po úspěšném otevření keše byl proveden pokus o opakované otevření stejné keše. V předchozím testování původních aplikací byla keš se stejným ID znovu otevřena. V implementaci byly provedeny změny, aby tato situace nenastala. Po pokusu o otevření stejné keše byla v mobilní aplikaci zobrazena hláška, která uživatele informuje, že již tuto danou keš otevřel. Výsledek testu je na obrázku 6.7 , kde vlevo je první otevření keše a vpravo druhé otevření stejné keše.

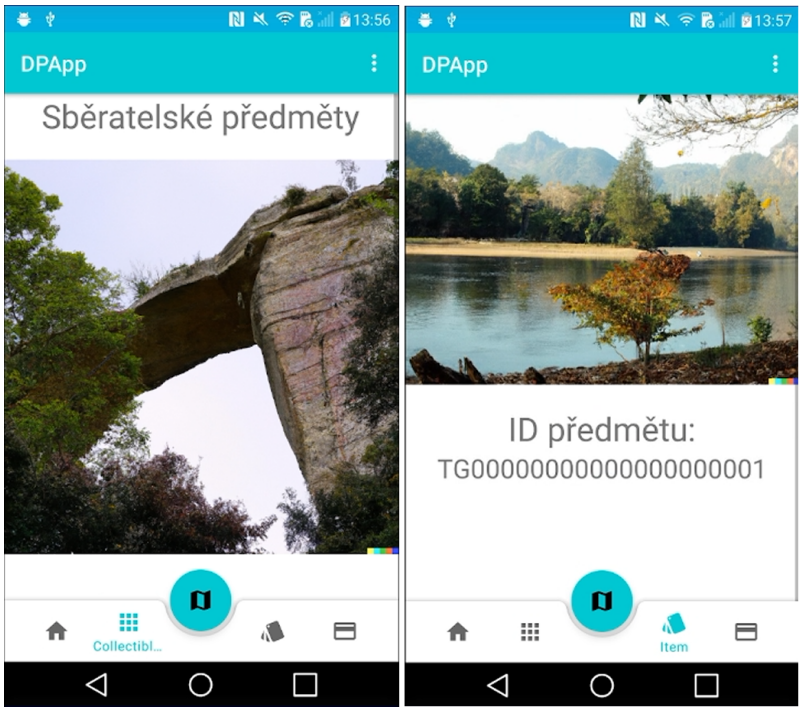


Obr. 6.7: Vlevo otevření keše, vpravo opakované otevření stejné keše.

Zároveň s otevřením keše proběhl i test toho, zda herní server správně odemkl sběratelský předmět. Vytvořený hráč, který byl použit pro otevření keše, měl všechny sběratelské předměty v základním stavu, tedy nenalezené. Po otevření keše byl jeden předmět nalezen a zobrazen v aplikaci. Podobným způsobem byl zkontrolován i putovní předmět, který byl také úspěšně přenesen k hráči a zobrazen v aplikaci. Výsledky testů jsou na obrázku 6.8.

Výsledky obou těchto testů byly zkontrolovány i pomocí webového rozhraní herního serveru. Po přihlášení byl počet nalezených keší roven jedné, což odpovídá jedné otevřené keši. Stejně tak byl nalezený sběratelský předmět viditelný i zde. Putovní předmět bylo také možné najít ve webovém rozhraní a bylo také možné zobrazit, kterými keši předmět prošel. Jelikož se předmět nacházel pouze na jedné keši, byla na mapě zobrazena pouze jedna keš. Z těchto výsledků je zřejmé, že mobilní aplikace správně komunikuje s herním serverem a také správně probíhá komunikace s kartou.

Posledním testem byl test mapy v mobilní aplikaci. Pro test byla otevřena mapa a bylo zkontrolováno, zda jsou na mapě zobrazeny keše a zda je zobrazena poloha uživatele. Výsledek tohoto testu je na obrázku 6.9.



Obr. 6.8: Vlevo sběratelské předměty, vpravo putovní předměty po otevření keše.

Obr. 6.9: Mapa v mobilní aplikaci.

Závěr

V práci byl popsán klasický geocaching, kdy vznikl a jak funguje. Byly také popsány různé typy keší. Dále byla v práci popsána architektura pro elektronický geocaching, jednotlivé prvky architektury a funkce těchto prvků. Pro jednotlivé prvky architektury byl proveden jejich návrh a vybrán vhodný software pro budoucí implementaci. Dále bylo provedeno testování již vytvořených aplikací [3]. Byla otestována funkčnost mobilní aplikace a appletu karty a bylo určeno, zda budou tyto aplikace převzaty celé, nebo jen jejich části. Pro applet karty bylo rozhodnuto, že bude převzat celý. Z mobilní aplikace bude převzata část kódu pro komunikaci s kartou.

V práci je poté popsána implementace jednotlivých částí infrastruktury. Nejdříve byly implementovány aplikace herního serveru. Pro klientskou aplikaci byly vytvořeny jednotlivé komponenty a služby. Celá klientská aplikace byla implementována ve frameworku Angular. V serverové aplikaci byly vytvořeny jednotlivé vstupní body a příslušná databáze. Byla implementována logika jednotlivých vstupů a definováno, jaké data mají vracet a v jakém formátu. Dále byla provedena implementace tokenů pro zabezpečení přístupu k databázi a implementace vytváření digitálních certifikátů. Pro serverovou aplikaci byl použit framework Django, databázový software PostgreSQL, a program openssl pro vytváření certifikátů. V mobilní aplikaci byly vytvořeny jednotlivé aktivity a fragmenty. V těchto aktivitách a fragmentech byla implementována komunikace s herním serverem a také komunikace s čipovou kartou. Kód pro komunikaci s čipovou kartou byl mírně upraven, aby bylo možné použít data získané z herního serveru. Applet karty byl jen mírně upraven. Úpravou byla implementace funkce, která umožňuje applet znovu spustit po proběhnutí protokolu. V aplikacích nebylo implementováno použití vytvořených certifikátů, jelikož původní software používá a byl implementován pro nestandardní certifikáty. Aplikace získávají data z certifikátů velmi specifickým způsobem, který je neslučitelný s nově vytvořenými certifikáty.

Jako poslední bylo provedeno testování vytvořených aplikací. Funkčnost serverové aplikace byla otestována pomocí aplikace Postman. Bylo otestováno přihlášení uživatele a získání tokenů, získání dat z databáze a funkčnost autentizace pomocí tokenů. V klientské aplikaci bylo otestováno získávání dat z databáze, ukládání tokenů po přihlášení a vytvoření putovního předmětu. V mobilní aplikaci bylo otestováno získávání dat z databáze a otevření keše. Po otevření keše bylo zkontrolováno, zda se příslušný putovní předmět správně přesunul a zda se odemkl sběratelský předmět. Obě tyto události proběhly úspěšně. Toto bylo zkontrolováno jak v mobilní aplikaci, tak i v klientské aplikaci.

Výsledkem práce jsou aplikace herního serveru, které obsahují databázi a umožňují vytvořit nového uživatele, putovní předmět a keš a také spravovat keše a putovní předměty uživatele. Dále je výsledkem mobilní aplikace, která umožňuje uživateli otevřít keš a vyměnit putovní předměty. Mobilní aplikace také umožňuje uživateli zobrazit informace o putovním předmětu a mapu keší. Aplikace z důvodu nekompatibility nepoužívají nově vytvořené certifikáty, ale původní certifikáty mobilní aplikace a appletu karty.

Literatura

[1] *Geocaching: Základní informace* [online]. Seattle: Groundspeak, c2023 [cit. 2023-04-27]. Dostupné z: <https://www.geocaching.com/sites/education/cs/frequently-asked-questions-cs>

[2] *Wiki Geocaching: Hlavní strana* [online]. 2020 [cit. 2022-10-08]. Dostupné z: <http://wiki.geocaching.cz/wiki/Hlavn%C3%AD_strana>

[3] Vertaľ D.: *Bezkontaktní mikropočítačová karta jako skrýš pro geokešing.* [Diplomová práce] VUT v Brně, Brno. Dostupné z: <https://bit.ly/3hgG7TR>

[4] *Wiki Geocaching: Historie* [online]. 2020 [cit. 2022-10-08]. Dostupné z: <http://wiki.geocaching.cz/wiki/Historie>

[5] *Geocaching: The History of Geocaching* [online]. Seattle: Groundspeak, Inc c2000-2022 [cit. 2022-10-08]. Dostupné z: <https://www.geocaching.com/about/history.aspx>

[6] *Geocaching: Fast facts* [online]. Seattle: Groundspeak, c2000-2022 [cit. 2022-10-08]. Dostupné z: <https://newsroom.geocaching.com/fast-facts>

[7] CARLY.: *What is Geocaching? GEOCACHING Official Blog* [online]. Seattle: Groundspeak, 2018, 5 Mach 2018 [cit. 2022-10-08]. Dostupné z: <https://www.geocaching.com/blog/2018/03/what-is-geocaching/>

[8] O'HARA, Kenton: *Understanding geocaching practices and motivations.* In: Proceedings of the SIGCHI conference on human factors in computing systems. 2008. p. 1177-1186 [cit. 2022-10-08]. Dostupné z: <https://dl.acm.org/doi/abs/10.1145/1357054.1357239>

[9] *Geocaching: Geocache Types* [online]. Seattle: Groundspeak, c2000-2022 [cit. 2022-10-08]. Dostupné z: <https://www.geocaching.com/about/cache_types.aspx>

[10] Ondra. *Přijde doba elektronických logbooků?.* Geocaching Shop U Tiskaře [online]. Jablonec nad Nisou, 2021 [cit. 2022-10-23]. Dostupné z: <https://utiskare.cz/stranka/elektronicke-logbooky>

[11] Burda, K. *Contactless Smart Card as a Cache for Geocaching.* International Journal of Computer Science and Network Security, 2021, roč. 21, č. 7, s. 205-210. ISSN: 1738-7906 [cit. 2022-10-22].

[12] Rescorla, E., *"The Transport Layer Security (TLS) Protocol Version 1.3"*, RFC 8446, DOI 10.17487/RFC8446, August 2018, [cit. 2022-10-22]. Dostupné z: <https://www.rfc-editor.org/info/rfc8446>

[13] LAWTON, George. *TechTarget: near-field communication (NFC).* In: TechTarget [online]. Newton (Massachusetts), c 2003-2022 [cit. 2022-10-22]. Dostupné z: <https://www.techtarget.com/searchmobilecomputing/definition/Near-Field-Communication>

[14] AWATI, Rahul a Peter LOSHIN. *Certificate authority (CA). TechTarget* [online]. Newton (Massachusetts): TechTarget, c 2000-2022 [cit. 2022-10-30]. Dostupné z: <https://www.techtarget.com/searchsecurity/definition/certificate-authority>

[15] WEISE, Joel. *Public key infrastructure overview.* Sun BluePrints OnLine, August, 2001, 1-27.

[16] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, *"Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile"*, RFC 5280, DOI 10.17487/RFC5280, May 2008, Dostupné z: <https://www.rfc-editor.org/info/rfc5280>.

[17] RUBIO, Daniel. *Beginning Django: Web Application Development and Deployment with Python.* Berkeley, CA: Apress, 2017. ISBN 978-1-4842-2787-9.

[18] Django: *Django documentation* [online]. Lawrence (Kansas): Django Software Foundation, c2005-2022 [cit. 2022-11-05]. Dostupné z: <https://docs.djangoproject.com/en/4.1/>

[19] *Django REST framework* [online]. Encode, c2011-2022 [cit. 2022-11-05]. Dostupné z: https://www.django-rest-framework.org/

[20] *Angular: What is Angular?* [online]. c2010-2022 [cit. 2022-11-06]. Dostupné z: <https://angular.io/guide/what-is-angular>

[21] BIERMAN, Gavin; ABADI, Martín; TORGERSEN, Mads. *Understanding typescript.* In: European Conference on Object-Oriented Programming. Springer, Berlin, Heidelberg, 2014. p. 257-281.

[22] *JSON Web Token Introduction* [online]. San Francisco (Kalifornie): Okta, c2022 [cit. 2022-11-27]. Dostupné z: <https://jwt.io/introduction>

[23] *Java Card 3 Platform Development Kit User Guide, Classic Edition* [online]. Austin (Texas): Oracle, c1998-2015 [cit. 2022-11-19]. Dostupné z: <https://docs.oracle.com/cd/E59935_01/guide/introduction.htm#JCUGC111>

[24] *Angular: Introduction to the Angular docs* [online]. Mountain View (Kalifornie): Google, c2010-2023 [cit. 2023-04-02]. Dostupné z: <https://angular.io/docs>

[25] ŠKRHÁK, P *Klientská aplikace* [cit. 2023-05-04] Dostupné z: <https://paavlex.github.io/dpfrontend-dist/>

[26] ENCODE. *Django REST framework. Django REST framework* [online]. Encode, c2011-2023 [cit. 2023-04-03]. Dostupné z: <https://www.django-rest-framework.org/>

[27] *Django documentation* [online]. Django Software Foundation, c2005-2023 [cit. 2023-04-03]. Dostupné z: <https://docs.djangoproject.com/en/4.2/>

[28] ŠKRHÁK, P *Serverová aplikace* [cit. 2023-05-04] Dostupné z: <https://dpapp.onrender.com>

[29] *Java Card v2.2.2: Package javacard.security* [online]. Santa Clara, CA: Sun Microsystems, c1993-2005 [cit. 2023-04-25]. Dostupné z: <https://www.win.tue.nl/pinpasjc/docs/apis/jc222/index.html>

[30] *Package javacardx.security.cert* [online]. Austin (Texas): Oracle, c1998-2021 [cit. 2023-04-25]. Dostupné z: <https://docs.oracle.com/en/java/javacard/3.1/jc_api_srvc/api_classic/javacardx/security/cert/X509Certificate.html>

Seznam symbolů a zkratek

GPS	Global Positioning System
NFC	Near Field Communication
TLS	Transport Layer Security
CA	Certifikační autorita
HS	Herní server
VK	Vlastník karty
K	Karta
HA	Hráčská aplikace
USB	Universal Serial Bus
HTTPS	Hypertext Transfer Protocol Secure
HTTP	Hypertext Transfer Protocol
AVP	Attribute–Value Pair
ECDSA	Elliptic Curve Digital Signature Algorithm
ASCII	American Standard Code for Information Interchange
R_S	Náhodné číslo karty
S	Číslo karty
Z_H	Zpráva hráče
P_H	Podpis zprávy hráče
U_H	Číslo putovního předmětu hráče
Z_S	Zpráva karty
P_S	Podpis zprávy karty
H	Číslo hráče
N	Pořadí hráče
U_S	Číslo putovního předmětu na kartě

PKI	Public Key Infrastructure
SQL	Structured Query Language
API	Application Programming Interface
REST	Representational State Transfer
JSON	JavaScript Object Notation
JWT	JSON Web Token
APDU	Application Protocol Data Unit
IDE	Integrated Development Environment
CLI	Command Line Interface
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets

Seznam příloh

A Specifikace protokolu pro diplomovou práci

Specifikace protokolu pro diplomovou práci Bc. Škrháka

Doc. K. Burda, CSc.

26.10.2022

1. Úvodní poznámky

Ke zvýšení atraktivity geokešingu se předpokládá možnost organizovat dva typy her. Prvním typem je putování virtuálních předmětů a druhým typem je sběratelství virtuálních předmětů. V případě putování virtuálních předmětů má každý hráč H možnost si vytvořit a u provozovatele herního serveru HS zaregistrovat svůj putovní virtuální předmět (například obrázek) s identifikátorem *U*. Putovní předmět se vyznačuje tím, že při každém nálezu skrýše se automaticky předává mezi smartfonem nálezce a skrýší. Ze smartfonu hráče H je tedy jeho putovní předmět předán do skrýše S, při nálezu této skrýše dalším hráčem X je tento putovní předmět předán do smartfonu hráče X atd. Putovní předmět tak postupně přes smartfony jednotlivých hráčů putuje různými skrýšemi po celém světě, což hráči mohou sledovat prostřednictvím herního serveru.

Druhým typem hry je sběratelství virtuálních předmětů. Provozovatel herního serveru definuje různé hry a hráč si ve své aplikaci zvolí, kterou z nich chce právě hrát. S každým nálezem skrýše pak hráčova aplikace vykoná jeden krok hry. Pokud se hráč rozhodl hrát skládačku, tak s každým nálezem nové skrýše mu herní aplikace zpřístupní nový dílek skládačky. Pokud hráč shromažďuje sérii nějakých virtuálních předmětů (např. obrázky historických aut), tak mu herní aplikace při nálezu skrýše zpřístupní předmět, který ve své sbírce ještě nemá.

2. Kryptografický protokol

K přenosu jednotlivých hodnot v níže uvedeném protokolu se požaduje použít formát AVP (Attribute–Value Pair). Každá přenášená hodnota (např. identita H, certifikát *CRT*, náhodná hodnota *R* atd.) se přenáší jako trojice $Y = (A, L, V)$. První bajt A reprezentuje typ hodnoty (např. Identita) a druhý bajt L udává celkovou délku trojice Y v bajtech. Zbytek $(L - 2)$ bajtů je samotná hodnota V (například textový řetězec dané identity). Číslování typů je zapotřebí volit, aby bylo systematické. Inspiraci k tomu lze nalézt např. v RFC 2865.

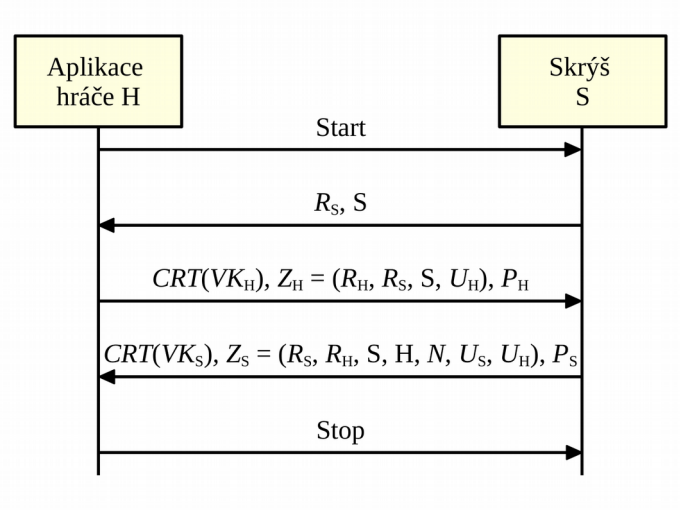
K podpisování zpráv požaduje použít techniku ECDSA 192 bitů a délka náhodných čísel musí být alespoň 32 bitů. Skrýš i hráčův smartfon mají v sobě bezpečně uložen veřejný klíč VK_{CA} certifikační autority.

Identifikátory jsou unikátním textovým řetězcem v kódování ASCII. Identifikátor skrýše je o délce nejvýše 12 bajtů, přičemž první dva bajty jsou znaky GC. Identifikátor putovního předmětu a hráče jsou řetězce o délce nejvýše 30 bajtů, přičemž první dva bajty identifikátoru putovního předmětu jsou znaky TG. Hráčův smartfon si vede seznam, do něhož ukládá identifikátory již nalezených skrýší, tj. skrýší, které mu vydaly potvrzení o jejich nálezu.

Nyní k samotnému běhu protokolu (viz obr. 1). Po ustavení přenosového kanálu mezi smartfonem a skrýší odešle hráčova aplikace zprávu Start, čímž zahájí běh protokolu. Skrýš pak protistraně odešle náhodné číslo R_S a svůj identifikátor S. Hráčova aplikace nejprve ve svém seznamu zkontroluje podle identifikátoru S, zda tato skrýš již nebyla hráčem nalezena. V kladném případě hráčova aplikace běh protokolu ukončí příkazem Stop a v opačném případě se v protokolu pokračuje. Uvedené opatření slouží k tomu, aby skrýš vydala každému nálezci jen jediné potvrzení o svém nálezu.

Smartfon poté skrýši odešle hráčův certifikát $CRT(VK_H)$ a dvojici Z_H, P_H, kde Z_H je hráčova zpráva a P_H je hráčův podpis této zprávy. Skrýš si nejprve pomocí veřejného klíče VK_{CA} certifikační autority ověří platnost hráčova certifikátu. Pokud je vše v pořádku, tak hráčův veřejný klíč VK_H z tohoto

certifikátu použije k ověření, zda podpis P_H je hráčův podpis zprávy Z_H. Jestliže je zmiňovaný podpis autentický, tak tím si skrýš ověřila identitu hráče H (jeho identifikátor zná z certifikátu) i autentičnost přijaté zprávy $Z_H = (R_H, R_S, S, U_H)$. V uvedené zprávě si skrýš ověří správnost jak hodnoty náhodného čísla R_S, tak i svého identifikátoru S a také si zapíše hodnotu R_H, což je náhodné číslo, které pro daný běh protokolu vygenerovala hráčova aplikace. Čísla R_H a R_S jsou vzhledem ke své náhodnosti pro daný běh protokolu unikátní a kontrola jejich hodnot znemožňuje útoky využitím dat, která byla přenesena v jiných běžích protokolu. Hodnota U_H je identifikátorem putovního předmětu, uloženém ve smartfonu hráče. Pokud je tato hodnota nula, tak to znamená, že se ve smartfonu hráče žádný putovní předmět nenachází.



Obr. 1: Kryptografický protokol pro komunikaci mezi aplikací hráče a skrýší

Skrýš poté vygeneruje potvrzení o jejím nálezu, což je zpráva $Z_S = (R_S, R_H, S, H, N, U_S, U_H)$, pro níž pomocí svého soukromého klíče vytvoří podpis P_S. Hodnota N je přitom pořadové číslo hráče H v postupnosti všech hráčů, kteří skrýš doposud objevili a hodnota U_S je identifikátorem předmětu, který je ve skrýši uložen. Pokud je tato hodnota nula, tak se ve skrýši žádný putovní předmět nenachází.

Následně skrýš odešle protistraně trojici $CRT(VK_S)$, Z_S, P_S. Hráčova aplikace nejprve pomocí veřejného klíče VK_{CA} certifikační autority ověří platnost přijatého certifikátu skrýše. Pokud je vše v pořádku, tak se z tohoto certifikátu použije veřejný klíč VK_S k ověření, zda podpis P_S je podpisem skrýše S pro zprávu Z_S. Je-li podpis autentický, tak tím si hráčova aplikace ověří identitu skrýše S i autentičnost zprávy Z_S.

Aplikace poté zahájí zpracování zprávy $Z_S = (R_S, R_H, S, H, N, U_S, U_H)$. V této zprávě nejprve ověří správnost hodnot obou náhodných čísel R_S a R_H a rovněž tak správnost identit S a H. Zpráva je prakticky potvrzením skrýše S, že hráč H je N–tým nálezcem této skrýše a v rámci tohoto nálezu došlo k výměně putovních předmětů tak, že ve skrýši je nyní uložen předmět s kódem U_H a ve smartfonu hráče se nyní nachází předmět s kódem U_S. Nakonec hráčova aplikace zašle skrýši příkaz Stop, čímž běh protokolu ukončí. Po přijetí tohoto příkazu skrýš inkrementuje hodnotu N a uloží si ji pro následujícího nálezce. Příkazem Stop může ukončit běh protokolu i skrýš. Tato situace nastane, když je negativní některá z kontrol dat přijatých z hráčovy aplikace. V takovémto případě je zapotřebí hráčův smartfon od skrýše nejprve oddálit a poté k ní opět přiblížit. Tím se spustí nový běh protokolu.

Hráčova aplikace ověří podpis P_S zprávy Z_S a správnost hodnot v této zprávě. Pokud je vše v pořádku, tak stanoveným způsobem aktualizuje stav hráčovy hry.

Poté se ještě uskuteční následující kroky, které však již nejsou součástí zadání práce. Hráčova aplikace přepošle přijatou trojici $CRT(VK_S)$, Z_S, P_S hernímu serveru HS. Herní server provede kontrolu zaslaného certifikátu i podpisu. Je-li vše v pořádku, tak hráče H zveřejní jako N–tého nálezce skrýše S. A pokud je hodnota U_S, resp. U_H nenulová, tak rovněž zveřejní, že hráč H nyní disponuje putovním předmětem U_S, resp., že ve skrýši S je nyní uložen putovní předmět U_H.

B Klientská aplikace

Přiložený archiv obsahuje zdrojové kódy klientské aplikace.

C Serverová aplikace

Přiložený archiv obsahuje zdrojové kódy serverové aplikace.

D Android aplikace

Přiložený archiv obsahuje zdrojové kódy android aplikace.

E Applet karty

Přiložený archiv obsahuje zdrojové kódy appletu karty, a také soubor *build.xml*.