Návrh souhrnné metodiky pro automatizované testování aplikací platebních terminálů na platformě Android

Bc. Marián Lenčéš

Diplomová práce
2021

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence

Akademický rok: 2020/2021

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Marián Lenčéš**
Osobní číslo:	**A17289**
Studijní program:	**N3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Forma studia:	**Kombinovaná**
Téma práce:	**Návrh souhrnné metodiky pro automatizované testování aplikací platebních terminálů na platformě Android**
Téma práce anglicky:	**A Proposal of a Comprehensive Methodology for the Automated Testing of Payment Terminal Applications on the Android Platform**

Zásady pro vypracování

1. Popište problematiku platebních terminálů a jejich testování.
2. Nastudujte a rozveďte problematiku tvorby automatizovaných testů aplikací na platformě Android.
3. Vyberte vhodné prostředky pro tvorbu automatizovaných testů platebních terminálů na platformě Android.
4. Navrhněte vlastní metodiku pro tvorbu automatizovaných testů platebních terminálů na platformě Android.
5. Vytvořte testovací sady a návrh otestujte.
6. Vyhodnoťte a prezentujte výsledky.

Forma zpracování diplomové práce: **Tištěná/elektronická**

Seznam doporučené literatury:

1. C. JORGENSEN, Paul. *Software Testing: A Craftsman's Approach*. 4th ed. New York: Taylor & Francis Group, 2014. ISBN 978-1-4665-6068-0.
2. D. CRAIG, Rick a Stefan JASKIEL. *Systematic Software Testing* [online]. Artech House, 2002. ISBN 1580535089.
3. DESIKAN, Srinivasan a Ramesh GOPALASWAMY. *Software Testing: Principles and Practice*. India: Dorling Kindersley, 2005. ISBN 978-81-7758-121-8.
4. O'MAHONY, Donald, Michael PEIRCE a Hitesh TEWARI. *Electronic Payment Systems for E-Commerce*. Second ed. London: Artech House, 2002. ISBN 1-58053-268-3.
5. RADU, Cristian. *Implementing Electronic Card Payment Systems*. London: Artech House, 2003. ISBN 2002033224.
6. MACHARLA, Pradeep. *Android Continuous Integration*. North Carolina, USA: Apress, 2017. ISBN 978-1-4842-2796-1.

Vedoucí diplomové práce: **Ing. Petr Žáček**
Ústav informatiky a umělé inteligence

Datum zadání diplomové práce: **15. ledna 2021**
Termín odevzdání diplomové práce: **17. května 2021**

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
DĚKAN
L.S.

doc. Mgr. Milan Adámek, Ph.D. v.r.
děkan

prof. Mgr. Roman Jašek, Ph.D. v.r.
ředitel ústavu

Ve Zlíně dne 15. ledna 2021

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně;
- byl/a jsem seznámen/a s tím, že na moji diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování diplomové práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne

Marián Lenčéš v.r.
podpis studenta

ABSTRAKT

Diplomová práce je zaměřená na tvorbu souhrnné metodiky pro automatizované testování platebních terminálů na platformě Android. Teoretická část je zaměřena na problematiku platebních terminálů, automatizované testování a výběr vhodných technologií pro následnou implementaci testovacích sad. Praktická část se zabývá návrhem vlastní metodiky, ukázkou vybraných testovacích sad z metodiky a vyhodnocení získaných výsledků.

Klíčová slova: platební terminál, testování, automatizace, Android, Appium

ABSTRACT

The master thesis is focused on the creation of a comprehensive methodology for automated testingof payment terminals on the Android platform. The theoretical part is focused on the issue of payment terminals, automated testing and the selection of suitable technologies for the subsequent implementation of test suites. The practical part deals with the design of own methodology, example of selected test sets from methodology and evaluation of the obtained results.

Keywords: payment terminal, testing, automation, Android, Appium

Tímto bych rád poděkoval svému vedoucímu Ing. Petrovi Žáčkovi za profesionální přístup, cenné připomínky a odborné vedení, které vedlo ke zdárnému dokončení této práce. Nakonec bych rád poděkoval své ženě za podporu, bez které bych práci nedokončil.

Prohlašuji, že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD

Platební terminály jsou na trhu již od roku 1979. Díky nim lze provádět bezhotovostní platební styk, který pomalu ale jistě nahrazuje platbu pomocí hotovosti. Pro banky, tak i pro jejich zákazníky je bezhotovostní platba vhodnější a méně riziková v porovnání s držením velkého obnosu peněz.

Dnešní svět jde technologicky rychle kupředu. Jako významný milník lze vidět příchod platebních terminálů se systémem Android na trh. Díky čemuž se nerozšířily jen funkce, ale i výkon, kterým terminály disponují. Tvůrcům platebních aplikací se otevřela nová možnost rozšíření svého portfolia.

Cílem diplomové práce bylo navrhnout souhrnnou metodiku pro automatizované testování aplikací platebních terminálů na platformě Android. Testování platebních terminálů postrádá veřejně dostupnou metodiku, ve formě základního doporučení, pro testování těchto zařízení. Výsledkem diplomové práce je návrh této metodiky.

Teoretická část se v první kapitole zabývá obecnými principy testování softwaru a základní terminologií dané problematiky. Druhá kapitola nastiňuje historii, architekturu a problematiku testování platebních terminálů. Třetí kapitola přibližuje platformu Android a její testování. Je zde také popsáno testování mobilních zařízení a s tím spojené problémy. Čtvrtá kapitola představuje testovací frameworky včetně jejich srovnání. Na základě srovnání jsou vybrány vhodné nástroje pro praktickou implementaci testovacích sad metodiky.

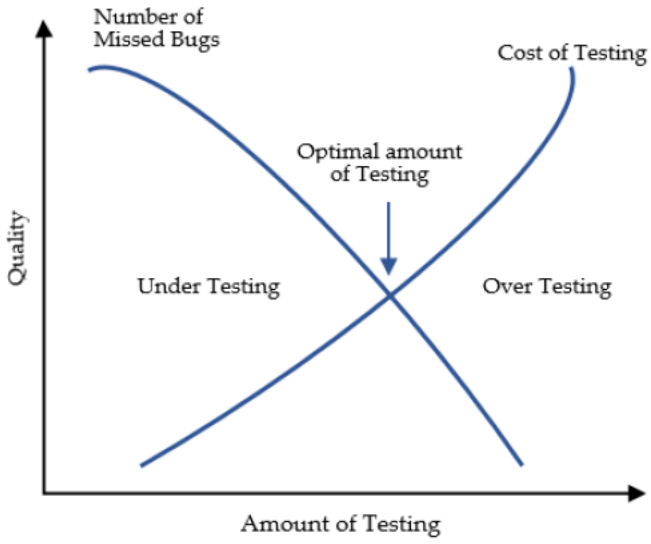
Praktická část diplomové práce se v páté kapitole věnuje návrhu metodiky pro tvorbu automatizovaných testů platebních terminálů na platformě Android a je rozčleněna dle testovacích úrovní. Šestá kapitola se zabývá doporučením vstupních a výstupních kritérií pro použití vytvořené metodiky. Sedmá kapitola obsahuje popis výběru vybraných testovacích sad včetně jejich logického členění v testovací případy. Testovací sady jsou vybrány s ohledem na pokrytí nejdůležitějších částí při testování platebních terminálů se systémem Android. Osmá kapitola se zabývá samotnou implementací testovacích sad, je zde prezentována ukázka implementovaných testovacích případů. Poslední kapitola se zabývá zhodnocením dosažených výsledků, které byly získány při běhu implementovaných testovacích sad. Výsledky jsou vhodně prezentovány.

I. TEORETICKÁ ČÁST

1 TESTOVÁNÍ SOTWARU

Testování lze definovat jako proces hodnocení, při kterém se kontroluje, zda původní systém splňuje původně stanovené požadavky. Hlavní roli zde hrají požadavky definované uživatelem. Testování softwaru odkazuje na hledání chyb nebo chybějících požadavků ve vyvíjeném systému nebo softwaru. Z toho vyplívá, že zúčastněným stranám poskytuje přesné znalosti o kvalitě produktu. [1; 2]

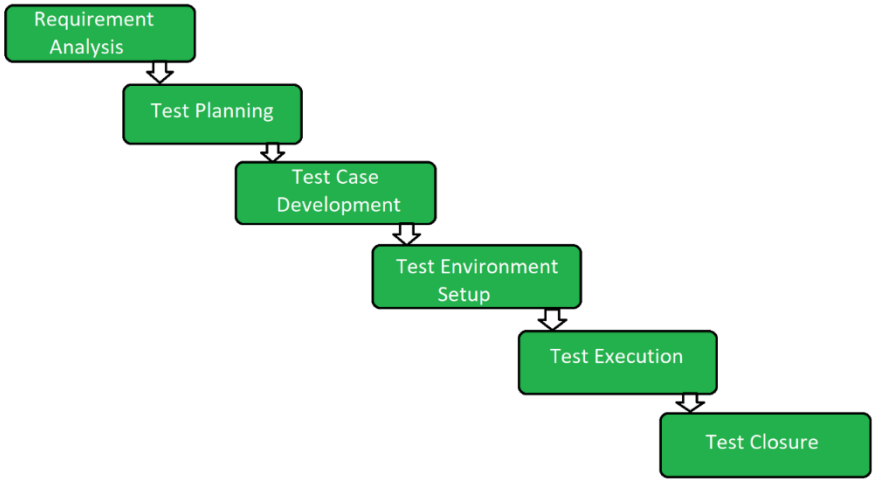
U testování softwaru je důležité vybírat vhodné testy do testovacích sad, aby se pokryla všechna rizika, které mohou nastat. Obrázek (Obr. 1) ukazuje náklady na testování ve vztahu s nalezenými chybami. Omezení testů a tím pádem ušetření prostředků může znamenat, že nebudou podchyceny všechny kritické chyby. Efektivním cílem testování je sestavit optimální množství testů tak, aby bylo možné minimalizovat další úsilí při testování. [1; 2]



Obrázek 1 Náklady na testování a nalezené chyby [2]

1.1 Životní cyklus testování softwaru

Životní cyklus testování softwaru se po světě liší různými úpravami. Obrázek (Obr. 2) popisuje fáze životního cyklu testování softwaru.



Obrázek 2 Životní cyklus testování softwaru [3]

V první fázi, tedy analýze požadavků životního cyklu, tým pro zajištění kvality (QA) provádí kontrolu požadavků na software.

Ve druhé fázi nastává plánování testů. Jedná se o nejdůležitější část životního cyklu. V této fázi jsou definovány všechny strategie testování a připravován testovací plán, který je výsledkem druhé fáze.

Třetí fáze se zajímá o vývoj testovacích případů. Testovací případy jsou zde detailně popsány a zkontrolovány. Dále zde probíhá příprava testovacích dat.

Čtvrtá fáze se zabývá nastavením testovacího prostředí. Testování prostředí je nezávislé na předchozí fázi a lze jej provádět zároveň s předchozím krokem. Testovací prostředí udává podmínky, při kterých je software testován.

Pátou fázi lze provést až po vykonání předchozích fází životního cyklu. Podle testovacích případů ze třetí fáze se zahájí testování. Pokud testovací případ projde bez chyby, označí se jako úspěšný. Neúspěšný testovací případ bude spojen s příslušnou chybou. Výstupem páté fáze je hlášení o chybě z neúspěšných testovacích případů, které jsou předané vývojovému týmu.

Poslední fáze životního cyklu se zabývá analyzováním procesu testování a jeho uzavření. [1]

1.2 Testovací techniky

Testovacích technik je v dnešní době značné množství. Základní rozdělení testovacích technik je na statické a dynamické. V následující časti jsou dané techniky detailněji popsány a rozvedeny. [4]

1.2.1 Statické techniky

Ve statických technikách testování jde hlavně o prevenci vad. Vyvinutý kód se zde nespouští. Statické techniky se mohou provádět dřív než dynamické techniky, a to už v rané fázi vývoje. Příkladem je kontrola dokumentů a statická kontrola kódu. Statickou kontrolu provádí zejména počítače pomocí různých nástrojů. Statické techniky nejsou pro vypracovávanou práci podstatné. [4]

1.2.2 Dynamické techniky

Při dynamických technikách testování se vyžaduje již spustitelná verze kódu. Systém se spustí se vstupy, poté proběhne a vygeneruje výstupy. Provede se kontrola běhu systému a jeho výstupu. Dynamické techniky testování lze rozdělit na black box, white box a gray box testování. Zmíněné druhy testování jsou rozebrány níže. [4]

1.2.2.1 Black box (černá skřínka)

Black box neboli testování černé skříňky je proces, při kterém jsou známe vstupní hodnoty a pokud je výsledek shodný s očekávaným výsledkem, tak test proběhl úspěšně. Hodnoty vstupu a výstupu jsou známé. Vnitřní fungování systému je skryto a není zde předmětem testování. [5; 6]

1.2.2.2 White box (bílá skřínka)

White box (clear box) neboli testování bílé skříňky je proces, při kterém nejsou známy jen vstupní a výstupní hodnoty, ale i způsob, jak se k danému výsledku došlo. Testování černé skříňky nemusí objevit chyby, které jsou známy jako náhodná správnost. To znamená, že pro zvolený vstup je výstupní hodnota správná, ale vnitřní proces k ní došel nesprávným způsobem a pro jiné vstupy může dávat nekorektní výstupy. Testování bílé skříňky by mělo takové chyby objevit. [5; 6]

1.2.2.3 Gray Box (šedá skřínka)

Gray box neboli testování šedé skříňky je proces, při kterém se kombinuje testování černé a bílé skříňky. [1]

Obrázek 3 White, Black and Gray box [3]

1.3 Úrovně testování

Každý software by měl projít všemi úrovněmi testování, než se uvolní do produkčního běhu. V ideálním případě se jedná o čtyři úrovně: komponentní, integrační, systémová a akceptační. Někdy může docházet i ke spojování jednotlivých úrovní a jejich výsledkem vznikají komponentně integrační nebo systémově integrační úrovně.

V praxi může docházet i k přeskočení integrační úrovně, kdy se software testuje přímo až na systémové úrovni.

1.3.1 Komponentní úroveň

Při komponentní úrovni se provádí takzvané testování jednotek, které ověřuje funkčnost konkrétní části kódu. U objektově orientovaného programování se jedná o testování jednotlivých tříd a metod.

Testy na komponentní úrovni vytváří sami programátoři, aby zajistili funkčnost jednotlivých částí kódu. Jedná se o white box testování. Jednotkové testy mohou odhalit chyby v raném stádiu vývoje. Lokalizování a oprava chyb je zde nejméně nákladná jak finančně, tak i časově. Testy lze spouštět i po jakékoliv změně v kódu.

Pomocí jednotkového testování lze odstranit konstrukční chyby před přechodem na integrační úroveň. [6]

1.3.2 Integrační úroveň

Při integrační úrovni dochází k propojování jednotlivých komponent, cílem je ověřit správnou funkčnost rozhraní mezi komponentami. K integrováním komponent lze přistoupit dvěma způsoby.

První způsob představuje postupnou integraci komponent, při které dochází ke snadné lokalizaci chyb a relativně rychlé opravě.

Druhý způsob představuje hromadnou integraci všech komponent a až pak následné testování rozhraní mezi komponentami.

Při integraci dochází k postupnému nabalování komponent, dokud nedojde k propojení všech komponent, které následně vytvoří výsledný systém. [6]

1.3.3 Systémová úroveň

Systémová úroveň je první úroveň, ve které se testuje celá aplikace jako celek. Testování provádí nezávislí testeři, kteří se nepodíleli na vývoji. K testování se přistupuje z pohledu zákazníka. Zkouší se scénáře, které v praxi mohou nastat. Součástí této úrovně jsou funkční a nefunkční testy. Samotné testování probíhá v prostředí, které úzce odráží produkci.

Systémová úroveň má za úkol prověřit, zda aplikace splňuje technické, funkční a obchodní požadavky stanovené zákazníkem. [6]

1.3.4 Akceptační úroveň

Cílem akceptační úrovně je ověřit splnění reálných potřeb a požadavků uživatelů. Poslední úroveň se provádí u koncových uživatelů podle předem domluvených scénářů. Dále sem patří zkušební běh aplikace. Pokud akceptační testování dopadne u zákazníka úspěšně, aplikace bude nasazena do reálného provozu. [6]

1.4 Automatizované testování

Pojmem automatizované testování lze vysvětlit jako testovací proces, nebo jeho část, prováděný bez přímého působení člověka, a to pomocí nástrojů, které jsou pro tyto účely speciálně vyvinuté.

Testování softwaru je časově náročná operace. Vývoj se urychluje a s tím je potřeba urychlit i proces testování. Zároveň s tím se zvyšují nároky na objem testů, které je potřeba pokrýt. Manuálním testováním by nebylo možné požadavkům vyhovět z pohledu ceny a rychlosti.

Díky tomu se v průběhu času začaly objevovat nástroje, které testerskou práci zefektivňují a snaží se jí usnadnit.

Testovací nástroje pro automatizované testování jsou charakterizovány vlastnostmi:

- Rychlost
    - Patří k největší výhodě automatizovaných testů. Při správně napsaných testech je nástroj pro automatizaci provádí několikanásobně rychleji. Jejich běh se provádí bez přestávek a pokaždé stejným způsobem. [7]
- Efektivita
    - Testovací nástroj provádí napsané testy testerem. Během této aktivity má tester čas na další pracovní úkony. Tester se opět zapojuje až po dokončení testů testovacím nástrojem, kdy kontroluje výsledky běhu. [7]
- Správnost a přesnost
    - U manuálního testování může docházet k lidské chybě. Může za to nepozornost, únava, neznalost či jiné faktory. Testovací nástroj provede testy vždy se stejným nasazením a stejným způsobem. Nemůže být ovlivněn faktory, které mohou nastat u člověka. [7]
- Neúnavnost
    - Běh testů lze provádět nepřetržitě bez přestávek. Lze je spustit v libovolnou hodinu a kolikrát chceme. [7]

V následující kapitole jsou lépe rozvedeny výhody a nevýhody automatizovaného testování. Pro lepší pochopení je zde provedeno srovnání i s manuálním testováním.

1.5 Manuální vs. automatizované testování

U automatizovaného testování existují mylné domněnky. Jedna z nich je, že při automatizovaném testování dojde ke snížení počtu testerů. Praxe udává opak. Ze začátku automatizace je potřeba většího počtů testerů. Po naprogramování všech testovacích scénářů se počet testerů vrací do původního stavu. Tito testeři nadále udržují a rozvíjí automatizované testy.

Automatizované testy jsou s manuálními testy úzce provázané. Při správném navrhnutí testovacího procesu by automatizované testy měly ulehčit manuálnímu testování. Manuální testování nelze vždy plně nahradit automatizovaným testováním. Některé testy by mohly být velmi obtížné na implementaci i přímo nemožné. V takové situaci se do automatizace daného testu neoplatí vkládat čas a finance. Automatizované testy se nejčastěji využívají

k regresnímu testování. Jedná se o testy, jenž se pravidelně opakují. Kontroluje se u nich, zda nedošlo s novou verzí aplikace nebo opravou některých chyb k rozbití funkcionality aplikace.

Automatizované testy mohou mít nedostatky v podobě intuice, kreativity a nepředvídatelných situací. Dělají jen to, na co jsou naprogramované. V takových situacích jsou manuální testy stále nenahraditelné. Tester se zamyslí nad danou problematikou a vyhodnotí vzniklou situaci.

V následujících částech jsou sepsány klady a zápory testování v manuální i automatizované formě. [1; 2]

1.5.1 Klady a zápory automatizovaného testování

Klady:

- Pokud jsou testy správně napsané, zjišťování místa problému je výrazně rychlejší.
- Nekonají se zde lidské chyby z nepozornosti. Vysoká spolehlivost.
- Rychlost provádění testů.
- Běh testů lze uskutečnit v libovolný čas.

Zápory:

- Kladené větší nároky na testovací tým při vytváření testů.
- Některé testy nelze automatizovat.
- Občasné problémy s detekcí objektů.

1.5.2 Klady a zápory manuálního testování

Klady:

- Vstupní náklady jsou výrazně nižší.
- Lze vytvářet složitější kombinace programů pro ověření výsledku.
- Testeři se snadněji přizpůsobují změně v aplikaci, která se testuje.
- Lze provést testy, které při automatizaci nejsou možné.

Zápory:

- Problematicky se stíhá větší objem testů. Nelze důkladně otestovat všechny funkcionality v rámci krátkého časového úseku.
- Chyba v podání lidského faktoru. Zde hraje velkou roli stres, únava nebo neznalost.

2 PROBLEMATIKA PLATEBNÍCH TERMINÁLŮ A JEJICH TESTOVÁNÍ

S platebními terminály se lze setkat například u obchodníků. Pro společnost jsou platební terminály samozřejmostí a nepřemýšlí nad nimi. Díky nim není potřeba, aby společnost disponovala určitým obnosem hotovosti. Život bez kreditních karet je již nepředstavitelný. Daná kapitola blíže seznamuje čtenáře se staršími terminály, které jsou aktuálně stále na trhu. Je zde uvedena i novější generace terminálů s operačním systémem Android s funkcemi, které přináší.

2.1 Historie platebních terminálů

Historie platebních terminálů je úzce spojená s historií platebních karet. Začátky spočívaly v kartičkách, které obsahovaly údaje o držiteli. Při platbě se dané údaje opisovaly. Tento proces byl značně zdlouhavý. V roce 1959 přišla společnost American Express s první plastovou kreditní kartou. Obsahovala několik údajů o držiteli embosovaným písmem. Jednalo se o jméno, adresu a jedinečné identifikační číslo. [8]

Tento nový druh kreditních karet přinesl možnost obchodníkům vytvářet otisk na papírové lístky pro banku, obchodníka a zákazníka jako doklad o koupi. Otisk kreditních karet se prováděl pomocí zip-zap stroje. Tento stroj vytvářel otisk kreditní karty na uhlíkový papír. Díky této metodě došlo ke zrychlení plateb v obchodě, protože nebylo potřeba údaje vypisovat ručně. Dnes je možné se s tímto strojem setkat už jen zřídka, někteří obchodníci ho využívají jako zálohu při výpadku elektronických platebních terminálů. [8; 9]

Otisk kreditních karet byl stále zdlouhavý proces. V roce 1970 proto přibyl magnetický proužek na kreditních kartách. Magnetický proužek byl vyvinut společností IBM a obsahoval jméno držitele karty, číslo karty, autorizační kód a datum vypršení platnosti karty. [8]

Ke čtení magnetických proužků bylo potřeba vytvořit elektronické platební terminály. S jejich příchodem se platební operace staly bezpečnější a bylo možné transakci přijmout nebo odmítnout přímo na místě. V roce 1973 přišla společnost Visa (původně American Express) s prvním elektronickým systémem autorizace transakcí, který propojoval obchodníky s datovým centrem v roce 1973.

Díky již existujícímu elektronickému systému byly transakce prováděny elektronicky. Ověření vlastníka karty bylo stále potřeba provádět telefonickou formou. Ke změně došlo v roce

1979, kdy společnost Visa představila elektrický terminál, který se spoléhal čistě jen na magnetický proužek. [8; 9; 10; 11]

Nový terminál od Visa přinesl výrazné snížení doby zpracování karet oproti otiskům nebo telefonickému ověření.

Další významný milník pro historii platebních terminálů byl rok 1983. Výrobce terminálů Verifone vydal svůj první platební terminál v sérii nazvané ZON. Ten připomínal větší kalkulačku se čtečkou magnetických proužků. A stal se základem pro většinu terminálů, které poté přišly na trh. [8; 9; 10; 11]

Platební terminály se stávaly stále dostupnější, jejich rozšíření po světě a funkcionalita nadále rostla. Postupně se vyvinuly do menších a uhlazenějších forem s většími displeji, bezdrátovými funkcemi na krátké i dlouhé vzdálenosti, a dokonce i s mobilními platebními funkcemi.

Platební terminály za pouhých pár desítek let změnily způsob, jakým spotřebitelé platí za zboží. Peníze se přesouvají do elektronické podoby, kdy lze platit již hodinkami nebo mobilním telefonem.

2.2 Platební terminály

Platební terminály jsou označovaný též jako POS terminály. Jejich hlavním cílem je komunikovat s platebními kartami za účelem vykonávání bezhotovostních plateb. V praxi to funguje tak, že terminál u obchodníka získá informace o platební kartě zákazníka a pošle je do banky k autorizaci. Po autorizaci banka převede finanční prostředky z účtu zákazníka na účet obchodníka.

Platební terminály v sobě ukrývají velké množství funkcí, o kterých běžní uživatelé nemají ponětí. Na obrázku (Obr. 4) je vyobrazena starší generace terminálu s očíslováním jednotlivých komponent, se kterým se lze běžně setkat v obchodech. Design platebního terminálu je u výrobců velmi podobný, jako u terminálu výrobce Ingenico.

Obrázek 4 Platební terminál Ingenico

Přední strana platebního terminálu vyobrazena na obrázku (Obr. 4) má označeno 5 hlavních komponent. První komponentou je tiskárna s termo kotoučkem pro tisk účtenky. Druhou je displej s bezkontaktní čtečkou platebních karet. Třetí komponenta označuje klávesy pro ovládání terminálu. Čtvrtá komponenta čtečku magnetických proužků na platební kartě. A poslední označuje čtečku čipu na platební kartě.

Zadní strana terminálu obsahuje slot pro SIM karty, napájecí konektor ze sítě, baterku a konektory pro komunikaci terminálu s okolním světem.

Pro bližší seznámení je potřeba si komponenty platebních terminálů a jejich roli přiblížit.

- **Displej** – zobrazuje výzvy pro obchodníka/zákazníka k zadání údajů (částka, která má být účtovaná, zadávání PINu a další).
- **Čtečka karet** – jedna z nejdůležitějších komponent, která poskytuje veškerá fyzická i logická rozhraní pro čtení platebních karet (magnetický proužek, čip a NFC pro bezkontaktní způsob plateb).

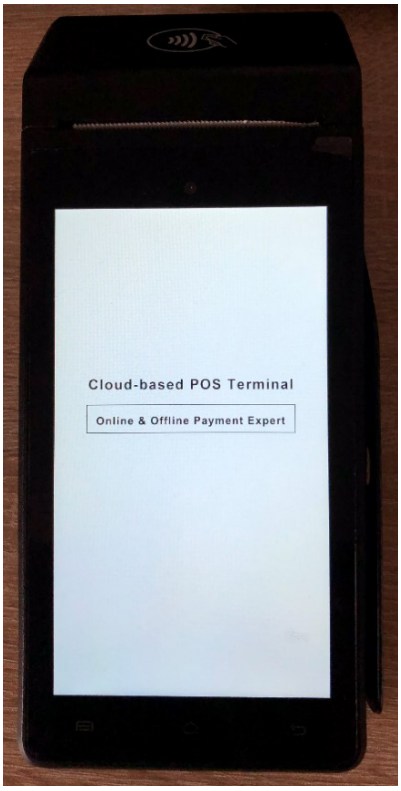
- **Tiskárna** – většina platebních terminálů obsahuje vestavěnou tiskárnu, která slouží pro tisk potvrzení karetních transakcí.
- **Klávesnice** – numerická klávesnice slouží k ovládaní platebního terminálu, nejčastěji je využívaná pro zadávání částky nebo PINu.
- **Externí hardwarové rozhraní** – externí rozhraní terminálu umožňují pro obchodníky autorizovat platby. Pro terminál je to jediný možný způsob, jak platby autorizovat a komunikovat s dalším externím příslušenstvím. Externí hardwarové rozhraní tedy poskytuje obousměrné připojení prostřednictvím různých fyzických (RJ11, USB) a síťových rozhraní (Ethernet, Wi-Fi, Celulární).
- **Platební aplikace** – jedná se o spojovací článek, který umožňuje zpracovat platbu. Platební aplikace řídí čtečku karet, displej a další rozhraní podle potřeb platebního procesu.

Nová generace platebních terminálů na platformě Android (Android POS terminály) jde oproti předchozí generaci výrazným skokem kupředu. Výkon terminálů narostl a s inteligencí se terminály přiblížily k dnešním chytrým telefonům. Nenarostl pouze jen výkon, ale i velikost obrazovek. [12]

Android POS terminály se staly velmi flexibilními zařízeními. Zmíněný nárust výkonu a velikost obrazovky umožnili vývojářům vytvářet složitější aplikace. Díky nim lze obchodníky lépe podporovat v jejich byznysu. Pokud se potřeby obchodníka v průběhu času změní, lze terminál upgradovat stažením nových aplikací podle jeho potřeb. [12]

Výhoda Android POS terminálů je v jejich širokém záběru použitelnosti. Terminály nemusí sloužit pouze jen pro vykonávání bezhotovostních plateb. Lze je využívat pro správu zaměstnanců, rezervace, správu zásob a další. To znamená, že Android APOS terminál nahrazuje více zařízení a služeb. Celkové náklady na vlastnictví takového terminálu budou výrazně nižší než vlastnit všechna zařízení, jenž terminál nahrazuje. [12]

Na obrázku (Obr. 5) je vyobrazena nová generace Android APOS terminálu. Oproti terminálu starší generace z předchozího obrázku (Obr. 4) je vidět značný krok kupředu a přiblížení se ke vzhledu mobilních telefonů.



Obrázek 5 Platební terminál Newland N910

2.3 Dělení platebních terminálů

Existují různé druhy platebních terminálů tak, aby pokryly co největší rozmanitost trhu a umožnily obchodníkům si vybrat takový druh terminálu, který bude sedět jejich požadavkům a druhu podnikání. V tabulce (Tab. 1) jsou rozděleny terminály na pět základních typů a ve stručnosti představeny jejich funkce a přednosti použití.

Tabulka 1 Rozdělení terminálů

Typy platebních terminálu	*Funkce*
Pultové platební terminály	Jsou ideální pro obchody, ve kterých zákazníci platí přímo u pokladny. Jedná se o terminály, které jsou připojeny k pokladně pomocí USB nebo Ethernet kabelu. Pultové platební terminály jsou řešeny buď samostatným terminálem nebo samostatným modulem PIN pad/čtečka karet/tiskárna.
Přenosné platební terminály	Přenosné platební terminály jsou ideální pro podniky, ve kterých se terminál přímo přinese za zákazníkem. Připojení k internetu je z pravidla přes Wi-Fi s dosahem až 100 metrů.

Mobilní platební terminály	Mobilní platební terminály jsou ideální pro obchodníky na cestách. Příkladem mohou být zaměstnanci rozvozové služby. Připojení k internetu je prostřednictvím mobilních sítí (4G, 3G, GPRS). Díky tomu lze terminál použít téměř kdekoli. Klíčovým faktorem mobilních platebních terminálů je životnost baterie.
Chytrý telefon/tablet a čtečka karet	Kombinace telefonu/tabletu a čtečky karet lze uplatnit u menších firem nebo i u obchodníků. Kombinace funguje stejně jako mobilní platební terminály. Mobil/tablet obsahuje software platebního terminálu a komunikuje pomocí mobilních sítí. Čtečka kreditních karet je připojena na mobil/tablet a může být součástí PIN padu nebo jako samostatný modul.
Virtuální platební terminály	Virtuální platební terminály reprezentují software na internetových stránkách, který umožňuje platby kreditní kartou.

2.4 Architektura platebních terminálů

Výrobce dodává platební terminály pouze s operačním systémem s různou úrovní programovatelnosti a přístupu k základním funkcím hardwaru terminálu. K terminálům dále poskytují SDK pro vývojáře aplikací, aby s jeho pomocí mohli vytvářet aplikace (POS systémy), které se po dokončení předkládají k certifikaci a schválení. Certifikované a schválené aplikace se následně mohou instalovat na terminály, které jsou určené do produkce k obchodníkům.

Pro lepší pochopení aplikací pro platební terminály, které zajišťují úspěšné provedení transakcí, je potřeba představit logickou architekturu platebních terminálu. Obrázek (Obr. 6) představuje grafické znázornění logické architektury platebních terminálů. U obrázku (Obr. 6) je dobré si popsat hlavní části architektury a představit je na příkladu. [13]

2.4.1 Vstupní bod

Vstupní bod zajišťuje zpracování konfiguračních dat, ověření a výběr vzájemně podporované aplikace, aktivaci odpovídajícího jádra, zpracování výsledku po dokončení jádra a předání výsledku čtečce.

Na obrázku (Obr. 6) je vstupní bod označen písmeny A, B, C a D. Vždy může začínat pouze na jednom z uvedených vstupních bodů. Záleží vždy na prostředí, tedy druhu vstupu platební karty, jestli dochází při transakci k restartu a další. [13]

2.4.2 Jádro

Jádro si lze představit jako software v POS systému, který má za úkol zpracovávat určitý typ transakcí. Existuje několik druhů jader a jejich volba probíhá podle priority a podle druhu platební karty. Platební karty od výrobců jako je VISA, MasterCard, American Express a další, potřebují vlastní jádro. K nim lze pro příklad přidat stravenkové karty a už se počet druhů jader dostává na 5 a více. Ve skutečnosti je jich daleko více. [13]

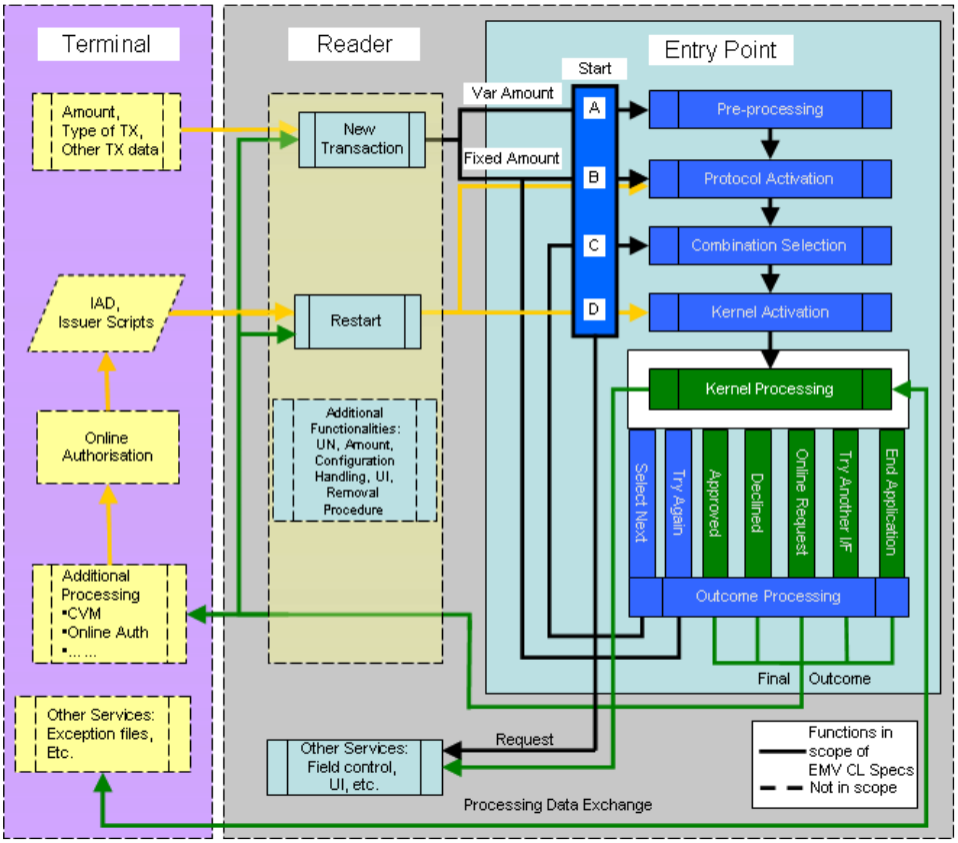
2.4.3 Výsledek

Výsledek představuje primární instrukci z jádra, která říká, jak by mělo zpracování pokračovat. Ovlivňuje například zprávy, které se mají zobrazit uživateli nebo informaci, zda se jádro bude restartovat po online autorizaci. [13]

Pro příklad lze zvolit transakci typu prodej. Jedná se o nejběžnější transakci na platebním terminálu.

V prvním kroku musí prodejce na terminálu zadat novou transakci. Na terminálu zadá částku, kterou má zákazník uhradit. Ve čtečce se vyvolá nová transakce, zákazníkovi se na displeji objeví částka k úhradě a řízení se předá vstupnímu bodu. Vstupní bod čeká na zákazníka, aby přiložil svojí kreditní kartu ke čtečce karet. Po přiložení platební karty získá vstupní bod potřebné informace o platební kartě. Díky těmto informacím a prioritám, které má nastavené, spustí odpovídající jádro a předá mu řízení. Po zpracování informací jádrem se řízení předá vstupnímu bodu. Na obrazovku zákazníka se vypíšou informace o zpracované transakci, zda byla úspěšná nebo ne.

V některých případech může být vyžadovaná i online autorizace. Ta je většinou spojena s částkou nad určitou hodnotu. Zákazník je v takové situaci vyzván k zadání PINu platební karty. PIN je následně zakódovaný do odeslaného autorizačního požadavku. Při online autorizaci dochází k restartování jádra a následnému vyhodnocení výsledku. Pokud byla autorizace schválena, zákazníkovi se transakce zobrazí jako úspěšná.



Obrázek 6 Logická architektura platebních terminálů [13]

2.5 Testování platebních terminálů

Platebními terminály se po světě zabývá řada firem. Každá firma k vývoji i následném testování přistupuje po svém. Know-how si jednotlivé firmy střeží, aby nepřišly o svoji výhodu na trhu.

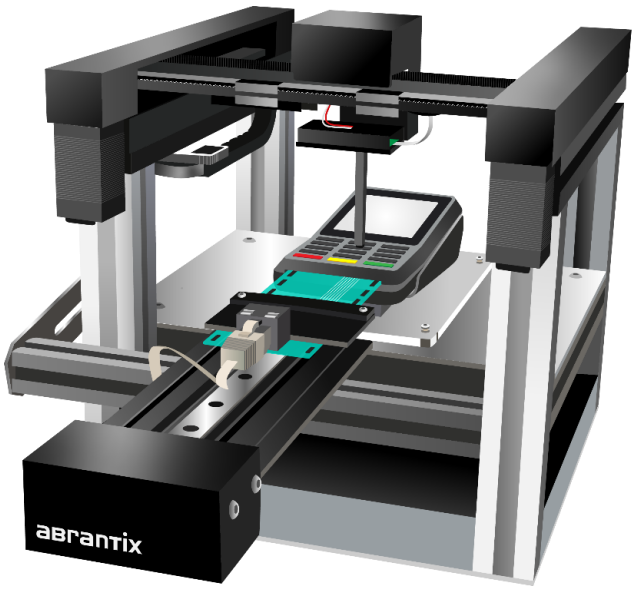
Starší generace platebních terminálů je na testování náročnější. Mnoho firem přistupuje k testování převážně manuální formou. Takový druh testování nemusí mít vždy shodné výsledky. Musí se zde zohledňovat lidská chyba.

Na obrázku (Obr. 4) byla představena starší generace terminálu. Výkon těchto terminálu je oproti novější generaci na platformě Android výrazně nižší. Často u nich chybí dotyková obrazovka, kterou nahrazují tlačítka. Automatizace je o to náročnější.

Software pro starší generaci platebních terminálů lze automatizovat v simulovaném prostředí, které je vytvořené na míru danému softwaru. Jeho nevýhoda je, že neběží

na platebním terminálu, ale na běžném počítači. Díky tomu se nedá plně spolehnout na výsledky z takto provedených testů.

Na automatizované testování platebních terminálů se specializují společnosti Abrantix a UL. Díky jejich partnerství vznikl automatizační testovací robot. Ukázka automatizovaného testovacího robotu je na obrázku (Obr. 7). [14]



Obrázek 7 Testovací robot od firmy Abrantix

[14]

Automatizovaný testovací robot zvládá zadávat PIN stisknutím tlačítek a dotykových obrazovek. U platebních karet dokáže obsloužit všechny vstupy platebních karet. Robot není postaven jen na určitý terminál, ale je kompatibilní s každým platebním terminálem. [14]

Nová generace platebních terminálů na platformě Android přináší další možnosti do automatizovaného testování. Automatizační testovací robot má poměrně vysoké vstupní náklady a pro rychlejší automatizaci je jich potřeba větší množství. Ve hře jsou i technologie, které pro začátek nepotřebují takové investice. Zmínka je o automatizačních frameworcích pro mobilní aplikace.

Díky narůstající oblibě mobilních zařízení na platformě Android, přibývá automatizačních frameworků, které dokážou otestovat různé aplikace běžící na Androidu. Aplikace pro platební terminály jsou oproti běžným aplikacím pro mobilní telefony do značné míry shodné. Rozdílů je zde víc než dostatek, proto se musí přistupovat k jedinečným řešením šitým přímo na míru dané aplikaci.

3 ANDROID A AUTOMATIZOVANÉ TESTOVÁNÍ

Android běží na velké škále zařízení. Nejčastěji na mobilních zařízení, které ovlivňují Android a běh aplikací svojí hardwarovou skladbou. Pro lepší uvedení problematiky automatizovaného testování na platformě Android, je potřeba si ve zkratce představit problematiku automatizovaného testování na mobilních zařízeních.

3.1 Android

Jedná se o open source systém, který běží na nespočtu zařízení po celém světě. Android se stal celosvětově nejúspěšnější platformu na mobilní telefonech. Neběží pouze na mobilních telefonech, ale lze se s ním setkat v televizích, hodinkách, tabletech a autech.

3.1.1 Historie

Počátek Androidu přichází se společností Android Incorporated v roce 2003. Vývoj probíhal utajeně. V roce 2005 přichází Google a odkupuje Android Incorporated. S tím přichází rozhodnutí použít Linux jako základ pro operační systém Android. Dané rozhodnutí přineslo možnost nabízet operační systém Android bezplatně výrobcům mobilních telefonů třetích stran. Příchod roku 2007 Google oznámení první otevřenou platformy pro mobilní telefony. Objevila se první Beta verze a v roce 2008 přišel Android 1.0 Apple Pie.

Dnes je Android nejrozšířenějším mobilním operačním systémem na světě. [15]

3.1.2 Architektura Androidu

Pro lepší pochopení Androidu je nutné uvést jednotlivé vrstvy, ze kterých se Android platforma skládá. Skládá se ze systémových aplikací, aplikačního rámce, nativní knihovny, Android Runtime, hardwarové abstraktní vrstvy a linuxového jádra. Všechny vrstvy jsou zobrazeny na obrázku (Obr. 8).

3.1.2.1 Linuxové jádro

Jak už bylo zmíněno výše, tak základem platformy Android je jádro Linuxu. Na linuxové jádro spoléhá vrstva Android Runtime. Ta využívá správu paměti na nízké úrovni nebo práci s vlákny.

Díky linuxovému jádru mohou výrobci zařízení vyvíjet ovladače pro hardware a Android klíčové funkce zabezpečení. [15; 16]

3.1.2.2 Hardwarová abstraktní vrstva

Poskytuje standardní rozhraní, pomocí kterého vystavuje hardwarové možnosti zařízení vyšší vrstvě aplikačního rámce. Hardwarová abstraktní vrstva se skládá z několika knihovních modulů. Každá hardwarová komponenta je přiřazena ke knihovnímu modulu. Pod hardwarovou komponentou si lze představit fotoaparát, GPS modul a další.

3.1.2.3 Android Runtime

Hlavním účelem Android Runtime je spouštění více virtuálních strojů na zařízení s minimální paměťovou stopou. Od Android verze 5.0 každá aplikace běží ve vlastním procesu. Tento proces obsahuje svou vlastní instanci Android Runtime.

S příchodem novějších verzí se Android Runtime postupně zdokonaluje. Důkazem je toho například lepší podpora ladění, podrobné diagnostické výjimky, hlášení o selhání nebo optimalizace sběru odpadu (garbage collection). [16]

3.1.2.4 Nativní knihovny

Nativní knihovny jsou napsané v programovacích jazycích C a C++. Systém Android je na nich závislý, protože mnoho základních komponent a služeb je vytvořeno z nativního kódu. Příkladem jsou hardwarová abstraktní vrstva nebo Android Runtime. [15; 16]

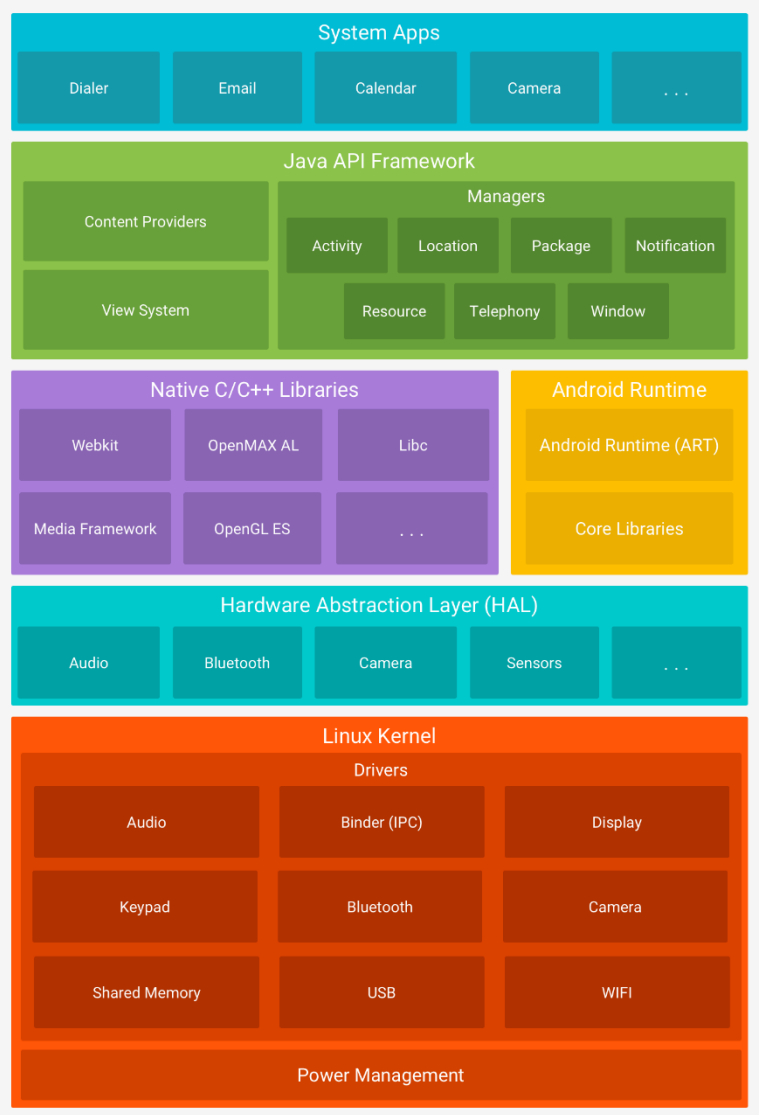
3.1.2.5 Aplikační rámce

Android vývojářům poskytuje stejné rozhraní API, jako mají systémové aplikace. Dále mají přístup ke všem existujícím knihovnám programovacího jazyka Java.

3.1.2.6 Systémové aplikace

Mezi základní sadu aplikací, která se dodávají s Androidem patří kalendář, zasílání SMS zpráv, kontakty, e-mail a další. Zmíněné aplikace nemají prioritní status, tedy nejsou upřednostňované oproti aplikacím třetích stran. Uživatel nemusí používat webový prohlížeč, jenž je předinstalovaný, ale může zvolit prohlížeč třetích stran, který bude pro něj na zařízení prioritní. Platí zde pár výjimek. Jedna z takových výjimek je aplikace nastavení systému.

Systémové aplikace lze využívat i jiným způsobem než jako běžný uživatel. Vývojáři je mohou využít jako klíčové funkce. To znamená, že vývojář do své aplikace nemusí implementovat funkcionalitu již obsaženou v systémových aplikacích. Příkladem toho je zasílání SMS, kdy vývojář smí ze své aplikace vyvolat aplikaci pro SMS a poslat přes ní zprávu. [15; 16]



Obrázek 8 Architektura systému Android [13]

3.2 Typy mobilních aplikací

Mobilní aplikace se dnes dělí do tří hlavních typů. Jsou to nativní aplikace, webové aplikace a hybridní aplikace. Ke každému typu aplikace se musí přistupovat jiným způsobem.

3.2.1 Nativní aplikace

Nativní mobilní aplikace jsou vyvíjeny pro konkrétní typ mobilního operačního systému. Aplikace se vytváří příkladem pro Android, iOS a další platformy. Konkrétní typ platformy říká, že aplikace vytvořená pro Android nemůže být používaná pro iOS. Platformy se mezi sebou nemohou kombinovat. Takové aplikace mají mnoho kladů ale i záporů. [17]

Mezi klady by se dala zařadit spolehlivost a rychlost z hlediska výkonu. Lépe dokážou hospodařit se zdroji zařízení a jsou schopny poskytnout optimalizovanější prostředí. Mezi další klady patří přístup k široké škále funkcí zařízení, jako je Bluetooth, fotoaparát, NFC a další.

Mezi zápory daného typu aplikací patří vynaložené úsilí k duplikování aplikace na jinou platformu. Kód vytvořený pro jednu platformu nelze znovu použít na jinou platformu. To s sebou pochopitelně nese zvýšení nákladů.

3.2.2 Webové aplikace

Jedná se o aplikace, jejichž chování je podobné jako chování u nativních aplikací. Webové aplikace se spouští pomocí webového prohlížeče. Při instalaci takové aplikace se ve skutečnosti instalace neprovede, ale dojde k uložení URL webu do zařízení. Pro vytvoření daného druhu aplikací se používá HTML5, JavaScript, CSS a další.

Mezi klady webových aplikací patří nižší nároky na paměť zařízeni, možnost přihlášení se z různých zařízení, nepotřebnost přizpůsobovaní se platformě. Dále dochází ke snížení nákladu na vývoj.

Mezi zápory lze uvést závislost aplikací na použitém webovém prohlížeči, nižší přístup k API pro vývojáře a horší funkcionalita v režimu offline. [17]

3.2.3 Hybridní aplikace

Většina hybridních mobilních aplikací jsou webové aplikace, které vypadají jako nativní aplikace. Výhody a nevýhody jsou kombinací webových a nativních aplikací. Vývoj těchto aplikací probíhá pomocí HTML5, CSS, Swift a další.

U hybridních aplikací bylo zmíněno, že mají výhody jak z webových, tak z nativních aplikací. Mezi hlavní výhody patří rychlost vytvoření aplikace. Tím pádem se zde ušetří čas a zdroje. Další výhodou je použití jednoho kódu pro více platforem.

Mezi nevýhody hybridních aplikací patří výkon, rychlost a celková optimalizace oproti nativním aplikacím. [17]

3.3 Automatizované testování na mobilních zařízeních

Mobilní zařízení v posledních letech zaznamenaly značný nárust výpočetního výkonu. Výkon se již v některých ohledech vyrovná stolním počítačům. S rostoucím výkonem jde ruku v ruce počet aplikací, které jsou pro mobilní platformu dostupné. Jejich rozsáhlost a propracovanost každým rokem stoupá. Testeři s aktuálním trendem musí držet krok, aby byli schopni dodat aplikaci řádně otestovanou a tím zajistit její kvalitu pro uživatele.

Automatizované testovaní desktopových aplikací je pro mnoho lidí shodné jako automatizované testování na mobilních zařízeních s Androidem. Jedná se stále o testování softwaru pouze na rozdílné velikosti obrazovky. Bohužel je zde mnoho rozdílností, díky kterým se musí na testování mobilních aplikací pohlížet jiným způsobem než na desktopové aplikace. [17]

Stěžejní oblasti při automatizovaném testování na mobilních zařízeních jsou zdroje zařízení, uživatelské vstupy, konektivita a bezpečnost.

Zdroje mobilního zařízení jsou oproti desktopovému zařízení omezené a musí se na ně dbát při vývoji a testování. Pro aplikaci si nelze zabrat všechny zdroje, které zařízení nabízí. Část zdrojů si zabírá operační systém, který na zařízeních běží a stejně tak jsou zde i aplikace na pozadí. Hospodárnost je doslova vyžadovaná. Pokud by aplikace zabírala nadměrné množství zdrojů, vlivem toho může dojít k zpomalování zařízení nebo pádu aplikace.

Mezi uživatelské vstupy lze řadit mikrofon nebo dotykovou obrazovku. Výjimečně se objevují zařízení s hardwarovou klávesnicí, ale těch je dnes opravdu minimum. V dnešní době jsou dotykové obrazovky hlavním prvkem, jejichž rozmanitost je na trhu značně velká, co se do rozlišení a velikosti týče. Uživatel přes dotykovou obrazovku zařízení ovládá. Tím se kladou větší nároky na GUI a jeho testování.

Konektivita zařízení se v průběhu času mění. Zařízení bývá s uživatelem v pohybu. Tím pádem může docházet ke změně síly signálu, k přepojení z mobilní sítě na Wi-Fi sít nebo k úplnému výpadku signálu zařízení. Aplikace by měla na vzniklou situaci umět reagovat a přizpůsobit se. [17]

Pokud aplikace pracuje s citlivými daty, je potřeba zajistit adekvátní zabezpečení k povaze aplikace. Spoléhat se na zabezpečení samotného zařízení je krátkozraké a nedostatečné.

Mobilní zařízení jsou silně personalizovaná zařízení, která obsahují data o uživatelích, které putují kamkoliv s uživatelem. Uživatel se může odkudkoliv připojit. Nejedná se o rozdíl

pouze v datech a použití zařízení, ale i v množství senzorů a možností vstupů, které mobilní zařízení umožňují. [17] V testování se na ně nesmí zapomínat, pokud je testovaná aplikace využívá. Spojení mezi testovanou aplikací a modulem je zásadní pro správnou funkčnost aplikace. Různé moduly vyžadují rozdílný přístup k testování. Do procesu testování se zapojuje větší kreace, pomocí které se hledá způsob, jak moduly otestovat.

Nejčastěji využívané moduly na mobilních zařízeních:

- Modul pro zjišťování polohy – GPS modul
- Síťový modul – 3G, 4G, 5G, EDGE, LTE
- Datový modul – Wi-Fi, Bluetooth
- Další moduly – NFC, kamera, G-senzor, světelný senzor a další

Desktopová zařízení většinu těchto modulů neobsahují. Při testování aplikací není potřeba na takové typy testů pomýšlet.

3.4 Problematika automatizovaného testování na platformě Android

Automatizované testování na platformě Android je rozdílné od dalších platforem jako iOS nebo Microsoft. Může za to otevřenost zdrojového kódu platformy Android.

Otevřenost softwaru zdrojového kódu Android má za následek, že si jej libovolný výrobce může upravit k obrazu svému a ušetřit tak náklady. Z pohledu výrobce je taková možnost velkým přínosem a promítne se to i do ceny zařízení. Bohužel z pohledu testera to přinese pouze starosti. Otevřenost kódu Androidu přinesla dvě hlavní problematiky:

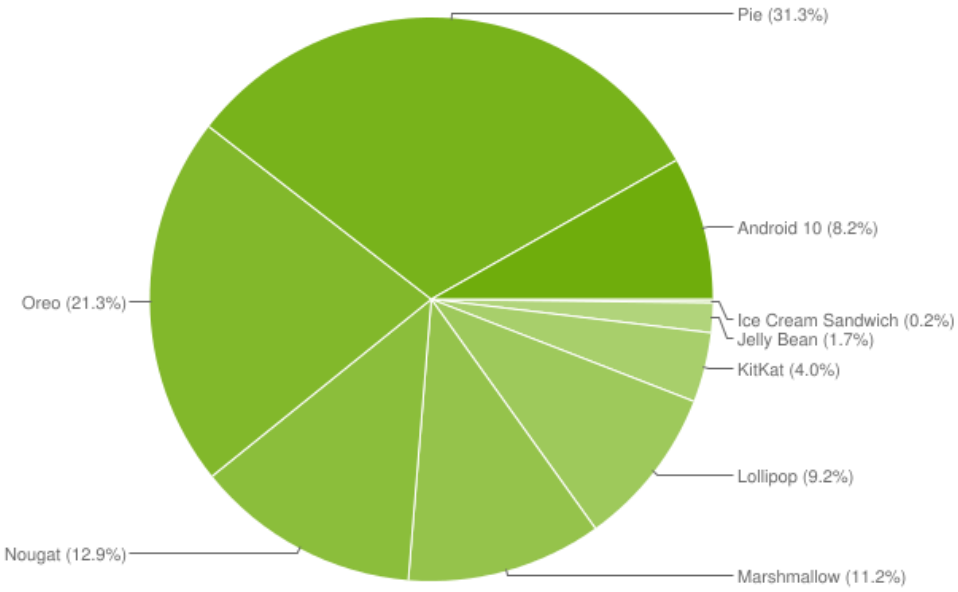
- Fragmentace hardwaru
- Fragmentace softwaru

Fragmentaci hardwaru přináší velké množství zařízení s rozdílným hardwarem, na kterých Android běží. Hlavní výhoda je ušetření nákladů, které by bylo potřeba vložit do vlastního vývoje výrobce zařízení. U levnějších zařízeních se pak lze setkávat s problematickým během systému, který přináší absenci plynulosti nebo nedostatečnou paměť zařízení.

Fragmentace softwaru přináší různé verze softwaru. Dané verze běží na různých zařízeních díky opožděným aktualizacím. Výrobci zařízení, kteří své zařízení osazují platformou Android nesou odpovědnost za aktualizace. Jak je již z praxe známo, aktualizace výrobce nemusí dodat nebo jich dodá jen omezené množství. Velký počet výrobců si nad platformu Android vytváří takzvanou grafickou nadstavbu. Pokud je grafická nadstavba špatně

odladěná, tak může způsobovat zpomalení chodu systému. Aktualizací mohou přímo zasahovat i do grafického rozhraní. Výrobce s aktualizací musí opravit nebo přepracovat grafickou nástavbu tak, aby s novými aktualizacemi fungovala správně. Díky této skutečnosti dochází k prodlevám ve vydávání aktualizací od výrobce zařízení.

Obrázek (Obr. 9) zobrazuje aktuální zastoupení různých verzí systémů Android na světě. Značné množství verzí přináší větší možnost chybovosti. Při vývoji aplikací je potřeba zajistit zpětnou kompatibilitu se staršími verzemi systému. Zde musí být kladen důraz na testování kompatibility, aby se otestovala funkčnost na všech podporovaných verzí systému.



Obrázek 9 Výskyt verzí platformy Android k první polovině 2020 [18]

3.5 Emulátory, simulátory a reálná zařízení

U mobilních aplikací lze využívat emulátory, simulátory nebo reálná zařízení. Bližší představení zmíněných možností je uvedeno v následujících odstavcích.

Emulátory mobilních zařízení jsou desktopové aplikace, které překládají mobilní aplikace tak, aby je bylo možné spustit na stolním počítači. Emulátory napodobují chování hardwaru a operačního systému mobilního zařízení. Emulátory nemohou nahradit mobilní zařízení, protože neobsahují všechny hardwarové prvky specifické pro mobilní zařízení. Mezi ně patří senzory.

Simulátory mobilních zařízení simulují malou podmnožinu chování hardwaru zařízení. Simulovaný hardware má vlastnosti pouze jako skutečné zařízení. Oproti emulátorům jsou podobné pouze cílové platformě. Stejně jako emulátory postrádají senzory a další. Díky úzkému vymezení jsou simulátory rychlejší než emulátory.

Hlavní rozdíl mezi simulátory a emulátory je v reprezentaci mobilního zařízení. Emulátor se snaží duplikovat celou vnitřní architekturu mobilního zařízení. Simulátor se naopak snaží duplikovat chování mobilního zařízení.

Reálné zařízeni reprezentuje mobilní telefon, tablet, televizi a další. Jedná se o skutečný hardware se senzory a dalšími komponentami.

Emulátor, simulátor nebo reálné zařízení poskytují různé výhody a nevýhody. Jejich shrnutí naleznete v následující kapitole.

3.6 Klady a zápory emulátorů, simulátorů a reálných zařízení

V předchozí kapitole byly stručně představeny prostředí, na kterých lze provádět manuální, ale i automatizované testy. V aktuální kapitole jsou rozvedeny výhody a nevýhody jednotlivých prostředí.

3.6.1 Klady, zápory emulátorů a simulátorů

Emulátory a simulátory spadají do stejné skupiny testovacích platforem. Uvedené klady a zápory jsou pro oba typy stejné.

Klady:

- Největší výhoda emulátorů a simulátorů je jednoznačně cena. Dodávají se s SDK a jejich použití je zdarma.
- Jejich využití je snadné.
- Vývojáři a testeři si mohou emulátory a simulátory konfigurovat podle potřeb. Příklad konfigurace může být zvolení verze operačního systému nebo rozlišení obrazovky a další.

Zápory:

- Mezi zápory lze zahrnout neobjevení chyb, ke kterým dochází jen na reálných zařízeních. Emulátory a simulátory nejsou přesnou kopií reálného zařízení. Ke zmíněnému faktoru nepřispívá ani možnost měnění uživatelského rozhraní výrobci

telefonů na platformě Android. Výrobci si uživatelské rozhraní přizpůsobují svým potřebám a tím dochází k rozdílnému chování uživatelského rozhraní u více výrobců s platformou Android.

- Mezi další nevýhody lze zařadit nemožnost napodobení hardwaru zařízení se všemi příslušnými senzory a rozhraními.
- Výkon CPU, GPU nebo paměť emulátoru a simulátoru neodpovídá reálnému zařízení.
- Síťové prostředí je simulováno.

3.6.2 Klady, zápory reálných zařízení

Klady:

- Výsledky testů jsou shodné s reálným používání.
- Možnost otestování skutečného výkonu CPU, GPU a paměti zařízení.
- S reálným zařízením můžeme testovat senzory zařízení.
- Lze testovat v reálných datových sítích.

Zápory:

- Jeden z největších záporů jsou náklady na koupi zařízení a jejich pravidelnou obnovu.
- Nelze testovat na všechny dostupných zařízeních z důvodu velkého množství.
- Časová náročnost údržby všech zařízení.

3.7 Automatizované testování na emulátoru, simulátoru nebo reálném zařízení?

Z uvedených informací lze vyvodit, že nejlepší výsledky v testování jsou na reálných zařízeních.

Testování na reálných zařízeních poskytují přesnější výsledky testů, pokud jde o spolehlivost, výkon a chování aplikace v reálném světě. Testy aplikace lze provádět na několika různých reálných zařízení současně. Testování na více zařízeních současně vede k urychlení času testování a odhalení nedostatků aplikace na daném zařízení.

V předchozím odstavci bylo doporučeno testování na reálném zařízení. Neznamená to ale, že by testování na emulátoru nebo simulátor bylo bezvýznamné. Především pro vývojáře

jsou emulátory a simulátory skvělou pomůckou. Poskytují rychlou zpětnou vazbu o aktuálním stavu mobilní aplikace. Jedna z největších výhod provádění automatizovaných testů nad emulátory a simulátory je ve škálování a paralelním zpracování, které je mnohem levnější a jednodušší než na reálných zařízeních. Ideálním řešením je najít vyvážený poměr pro emulátory, simulátory a reálná zařízení abychom dostali to nejlepší z jednotlivých prostředí. [17]

4 TECHNOLOGIE PRO AUTOMATIZOVANÉ TESTOVÁNÍ PLATEBNÍCH TERMINÁLŮ

Před několika lety byla automatizace mobilních zařízení značně problematická. Frameworky nedokázaly automatizovat testy pro více operačních systému v rámci jednoho projektu. Bylo potřeba využívat rozdílné nástroje pro různé operační systémy. Příchodem multiplatformních frameworků se situace změnila a dnes lze vytvořit automatizované testy, které lze provádět na mobilním zařízení s Androidem, iOS, ale i na počítači s Windows nebo Linuxem.

Před představením zvolených frameworků je potřeba si uvést primární kritéria jejich výběru:

- Programovací jazyk Java
- Spolehlivost
- Rychlost
- Rozšiřitelnost
- Cena

4.1 Appium

Appium je podle oficiálních stránek definován jako „*open-source automatizační framework pro použití s nativními, hybridními a mobilními webovými aplikacemi. Řídí aplikace pro iOS, Android a Windows pomocí protokolu WebDriver*.“ [19]

Z výše uvedeného popisu je zřejmé, že se jedná o open-source nástroj. Lze ho využít pro automatizaci nativních, mobilních webů a hybridních aplikací na mobilních platformách Android, iOS a na Windows desktopových platformách. Jedná se o multiplatformní nástroj, který umožňuje psát testy na více platformách pomocí stejného API. Díky tomu lze opětovně použít kód mezi testovacími sadami pro iOS, Android a Windows. [19]

Appium je založeno na Seleniu. Většina příkazů, které lze použít v Seleniu, funguje i v Appiu. Za předpokladu, že příkazy mají smysl pro mobilní automatizaci. [19]

Appium architektura umožňuje použít velké množstvím programovacích jazyků. Mezi ně patří Java, Python, JavaScript, Ruby, PHP a další. Architektura Appia je dále rozvedena v částí Appium architektura.

4.1.1 Podpora platforem

Podpora pro automatizaci iOS je tvořena pomocí dvou ovladačů:

- XCUITest
- UIAutomation

UIAutomation je již zastaralý ovladač. Používá se pro iOS 9.0 až iOS 9.3. Naopak XCUITest je novější ovladač a lze jej používat od iOS 9.3 až po nejnovější verzi. Appium zpravidla poskytuje podporu dvou posledním verzím iOS. [19]

Testování lze provádět na simulátoru i na reálných zařízení jako je iPhone, iPad a tvOS.

4.1.1.1 Podpora Androidu

Podpora pro automatizaci Androidu je pomocí dvou ovladačů:

- UiAutomator2
- UiAutomator

UiAutomator je již zastaralý ovladač. Appium podporuje Android již od verze 4.3. Testy lze provádět na emulátorech od Androidu a na reálných zařízení s Androidem. [19]

4.1.1.2 Podpora Windows

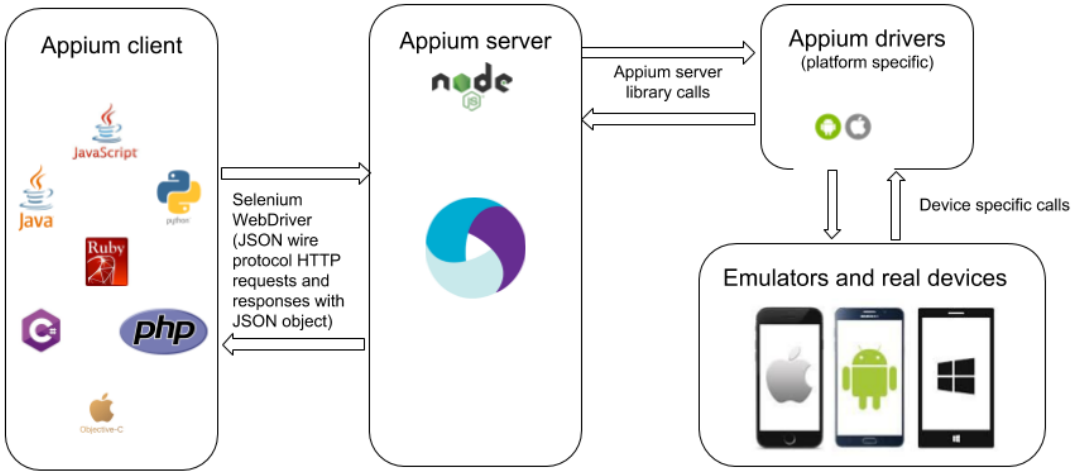
Podpora pro automatizaci Windows je zajištěna pomocí WinAppDriver ovladače. WinAppDriver umožňuje testování aplikací na Universal Windows Platform (UWP) a Classic Windows (Win32). [19]

Testy lze provádět na počítačích s Windows 10 a novější. Dále umožňuje vstoupit do režimu správce.

4.1.2 Appium architektura

Jedná se o architekturu klient/server, kde Appium je jádrem webového serveru, který zpřístupňuje REST API. Klient a server používají objekty JSON (JavaScript Object Notation) s předdefinovanými vlastnostmi při popisu funkcí. Vzhledem ke zmíněné architektuře lze k napsání příslušných HTTP požadavků na server použít klienta napsaného v jakémkoliv programovacím jazyku. Pro populární jazyky již klienti existují. [20]

Na obrázku (Obr. 10) je vyobrazená komunikace mezi klientem, serverem a zařízením. Server má za úkol přijímat připojení od klienta a naslouchat příkazům. Provádí tyto příkazy na mobilním zařízení nebo emulátoru a zasílá HTTP odpověď s výsledkem příkazů. [19]



Obrázek 10 Komunikace mezi Appium serverem, klientem, ovladači a zařízeními [21]

Klient předává serveru informace o operačním systému, verzi operačního systému, jedinečný identifikátor zařízení (UDID), jméno balíčku aplikace a další. Podle dodaných informací od klienta server zvolí příslušný ovladač, zařízení a aplikaci. Následně provede operace, které jsou napsané v testovacím skriptu. Výsledek těchto operací předá klientovi.

4.2 TestProject

První myšlenky pro zrození TestProject přišly v roce 2015. V dubnu 2018 se objevila první veřejná verze. TestProject přitom přinesl nové prvky do automatizovaného testování. Na oficiálních stránkách je označován jako platforma nové generace. [22]

TestProject je end-to-end automatizační nástroj, který poskytuje funkce pro automatizaci webových, API a mobilních aplikacích. Licence je open-source, tedy stejná jako v předešlých nástrojích. Stejně jako Appium, lze TestProjekt využít pro automatizaci nativních, mobilních webů a hybridních aplikací na mobilních platformách Android, iOS a na Windows, Linux a macOS desktopových platformách. Z toho vyplývá, že se jedná o multiplatformní nástroj. [22]

4.2.1 TestProject architektura

TestProject je postavený na Appiu a Seleniu. Odtud plynou multiplatformní vlastnosti. Ke zmíněným technologiím má velmi blízko a díky tomu nabízí nástroje na import testů napsaný v Appiu nebo Seleniu. [22]

TestProject se skládá ze dvou různých komponent pomoci kterých může uživatel vytvářet testy. Jedná se o Smart Test Recorder a TestProject SDK (Software development kit). Test Recorder umožňuje uživateli zaznamenávat testy z uživatelského rozhraní bez psaní kódu.
SDK umožňuje uživateli programovat jednotlivé testovací případy.

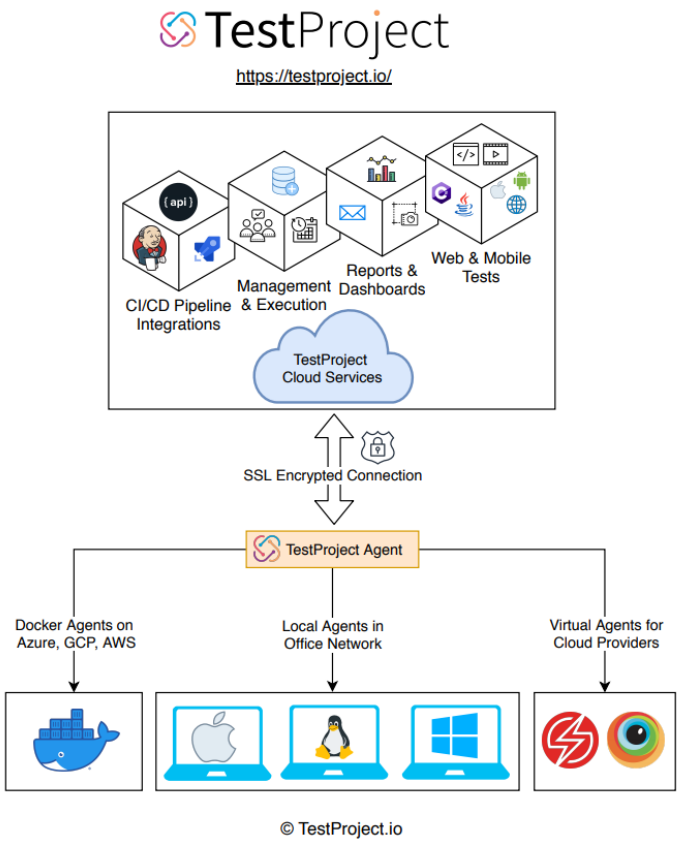
4.2.1.1 Smart Test Recorder

Automaticky zaznamenává a vytváří automatické testovací kroky při procházení uživatelského rozhraní aplikace. Pro nezkušené testery je to přínos, kdy i bez znalosti programování mohou vytvářet testy a postupně pronikat do světa automatizovaného testování. [22]

4.2.1.2 TestProject SDK

SDK je postaveno na Appiu a Seleniu. Syntaxí vychází ze zmíněných frameworků, aby usnadnil automatizaci webových a mobilních aplikací. Podporuje i veliké množství programovacích jazyků, mezi které patří Python, Java, C# a JavaScript. [22]

Na obrázku (Obr. 11) je graficky znázorněná architektura TestProject. Můžeme si ji rozdělit na dvě hlavní části. V první části jsou cloudové služby, které obsahují uložiště, ověřování uživatelů, správu a další. Ve druhé části jsou TestProject agenti, kteří umožňují vytvářet a provádět testy proti cíleným aplikacím. [22]



Obrázek 11 Architektura platformy TestProject [22]

4.2.2 TestProject Cloudová Platforma

Cloudová platforma je něco jako GUI pro TestProject. Uživatelé se zde mohou připojit, spravovat jednotlivé projekty, přidávat nebo odebírat testovací případy a plánovat úlohy. Jednotlivé projekty mohou být sdíleny v týmu. Díky tomu lze spolupracovat na daném projektu s ostatními členy týmu, kteří mohou projekt upravovat.

Cloudová platforma nabízí přístup k doplňkům, díky kterým lze zefektivnit automatizované testování. Kromě toho poskytuje integraci s nástroji pro analýzu a reportování.

4.2.3 TestProject Agenti

Agenti slouží jako komunikační most mezi zařízením a platformou TestProject. Také komunikují s cloudovou platformou pomocí zabezpečeného šifrovaného připojení SSL. Po připojení na cloudovou platformu, agenti mohou vykonávat všechny testovací případy na místním počítači.

4.3 Robotium

Zakladatel Redas Rana chtěl jako ostatní vytvořit funkční, robustní a snadně ovládající framework. Uživatelé by nepotřebovali hlubší znalosti o testované aplikaci. S tvorbou vlastního frameworku začal 10. ledna 2010. Postupem času začali přibývat nové funkcionality a Robotium se tak rozvíjelo do aktuální podoby.

Snaha Redase Rana se vydařila a vznikl open-source testovací framework pod licencí Apache verze 2.0. Poskytuje plnou podporu testování nativních a hybridních aplikací Android API od verze 8. Nižší nároky na testovanou aplikaci ulehčují a urychlují psaní testů pro black box testování GUI na platformě Android. Robotium bylo zkonstruováno tak, aby dokázalo pokrýt funkční, systémové a akceptační testovací scénáře napříč mnoha aktivitám jedné aplikace. Vytvořené testy lze provádět na emulátorech i na reálných zařízeních.

K Robotiu patří ještě takzvaný Robotium Recorder. Ten umožňuje rychlé nahrávání testovacích případů. [23; 24]

4.4 Selendroid

Selendroid představuje dalšího zástupce z open-source testovacích frameworků. Stejně jako předchozí testovací nástroje využívá licenci Apache 2.0. Selendroid umožňuje vytvářet testy pro nativní, hybridní aplikace pro platformu Android a mobilní web. Testování lze provádět na více reálných zařízeních nebo na více emulátorech současně. Platformu Android podporuje od verze 10. [24]

4.4.1 Selendroid architektura

Selendroid se skládá ze 4 hlavních komponent:

- Web Driver Client – klientská knihovna založená na Seleniu
- Selendroid-Server – server, který běží na testovaném zařízení nebo emulátoru
- Android Driver-App – integrovaný ovladač pro Android
- Selendroid-Standalone – komponenta jenž zajišťuje instalaci Selendroid serveru a testované aplikace

Stejně jako Appium využívá JSON wire protocol pro komunikaci mezi klientem a serverem. Server následně využívá integrovaný ovladač pro vykonání testů na testované aplikaci.

Selendroid je založen na Seleniu stejně jako již některé zmíněné testovací frameworky.

4.5 Zhodnocení testovacích frameworků

V tabulce (Tab. 2) jsou jednotlivé frameworky porovnány, tak aby byly zřejmé jejich výhody a nevýhody.

Tabulka 2 Porovnání testovaných frameworků

	Appium	*TestProject*	*Robotium*	*Selendroid*
Podporované platformy	Android, iOS, Windows	Android, iOS, Windows	Android	Android
Podporované verze Androidu API	Od verze 18 a výš	Od verze 21 a výš	Od verze 8 a výš	Od verze 10 a výš
Podporované druhy aplikací	Nativní, hybridní a webové	Nativní, hybridní a webové	Nativní a hybridní	Nativní, hybridní a webové
Podporované programovací jazyky	Java, Python, JavaScript, Ruby, PHP a další	Java, C#, Python	Java	Java, C#, Perl
Paralelní zpracování testů	Ano	Ano	Ne	Ano
Licence	Apache 2.0	Apache 2.0	Apache 2.0	Apache 2.0

Z tabulky (Tab. 2) vyplývá, že všechny vybrané testovací frameworky podporují programovací jazyk Java. Splňují také podmínku nízkých pořizovacích nákladů a jedná se o testovací frameworky s open-source licencí Apache 2.0.

TestProject splňuje požadavky, které byly na začátku kapitoly definovány. Po jeho implementaci bylo objeveno několik komplikací. Zmíněné komplikace jsou:

- Při představování TestProjectu byla zmíněna cloudová platforma. Ta umožňuje členům týmu si navzájem sdílet projekty. Právě sdílení projektů ze začátku nefungovalo podle očekávání. Následně se daný problém vyřešil vytvořením dalšího účtu u každého z členů týmu.
- Zpracování dokumentace nedosahuje přehlednosti a podrobnosti informací jako u jiných testovacích frameworků.

- Avizovaná rychlost frameworku nakonec nebyla uspokojivá.
- Problematická komunikace s podporou při řešení nastalých problémů.
- Plugin pro Cucumber nebyl v době výběru implementovaný.

Tyto komplikace vedly k zavrhnutí využívání TestProjectu.

Robotium proti ostatním frameworkům nepodporuje webové aplikace a paralelní zpracování testů. V případě většího množství testovacích scénářů bude potřeba víc času pro jejich vykonání. Navíc testovací scénáře nelze spouštět na více zařízeních zároveň. To nám již naznačuje, že se Robotium hodí pro menší projekty. Poslední aktualizace proběhla v roce 2016.

Selendroid je další testovací framework, který počáteční podmínky splňuje. Jeho přednost je při multiplatformním testováním s opakovatelně použitelným kódem pro testovací sady na Windows, Android a iOS. Poslední aktualizace Selendroidu proběhla v roce 2015. Stejně jako Robotium obsahuje řadu nevyřešených chyb. Opravy se již nevydávají, a proto byl zavrhnut z výběru.

Appium je poslední zmíněný automatizovaný testovací framework. Má řadu předností i negativ. Podstatnou výhodou Appia je, že se skvěle hodí k paralelnímu regresnímu testování velkých aplikací, u kterých dochází k průběžným úpravám a rozšíření funkcionalit. Aplikace běžící na platebním terminálu je poměrně rozsáhlá a dochází zde k rozšíření funkcionalit pro určité skupiny zákazníků. Propojení Appia s dalšími technologiemi je většinou snadné a bezproblémové. Během testování bylo Appium stabilní a dosahovalo očekávaných výsledků vzhledem k plánovaným testovacím sadám. Jeho podobnost se Seleniem přispívá k rychlému přeučení automatizovaných testerů. Appium nejlépe splňuje počáteční podmínky a jeví se nám jako nejvhodnější testovací nástroj.

4.6 Technologie rozšiřující testovací frameworky

Frameworky pro automatizované testování na platformě Android představují kvalitní základ, na kterém se dále může stavět. Pro dosažení požadovaného cíle, je potřeba přidat další technologie.

Aby testovací scénáře byly v projektu na první přečtení srozumitelné, lze použít často využívaný Cucumber. Testování je postaveno na objektově orientovaném programovacím jazyku Java. Pro další možnosti běhu automatizovaných testů bylo rozšíření o TestNG logickým krokem.

4.6.1 Cucumber

Cucumber je open source testovací přístup, který podporuje BDD a je napsaný v programovacím jazyce Ruby. Chování aplikace je popsáno pomocí jazyka Gherkin.

Cucumber vznikl po mnoha neúspěšných pokusech. Zrodil se v roce 2008 a jeho otcem je Alsak Hellesøy. „*Zrození Cucumberu předcházela frustrace s nejednoznačnými požadavky a nedorozumění mezi lidmi, kteří si software objednávají, a těmi, kteří jej dodávají.*“ (Hellesøy, 2014) [25]

Velké oblíbenosti Cucumberu přispívá podpora celé řady programovacích jazyků. Mezi nejznámější patří PHP, Ruby, Java, .NET, Node.js, Python a další.

4.6.1.1 Gherkin

Jedná se o sadu gramatických pravidel, díky nimž je text dostatečně strukturovaný a Cucumber mu dokáže porozumět.

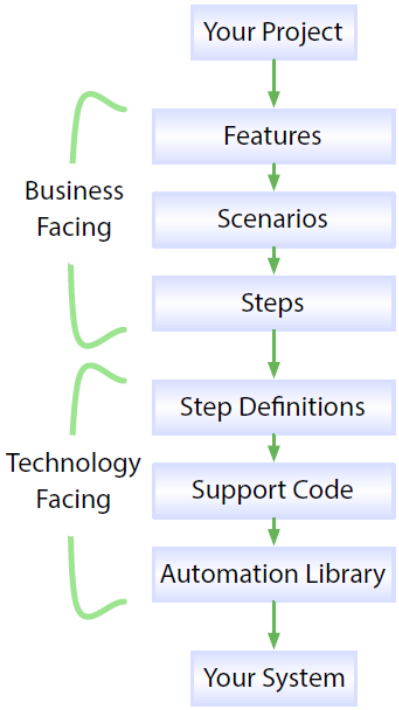
Gherkin obsahuje sadu speciálních klíčových slov. Každé klíčové slovo může být přeloženo do zvoleného jazyku. Scénáře není potřeba psát jenom v angličtině, ale jde zvolit řada jiných jazyků. [26]

4.6.1.2 Cucumber architektura

Pro porozumění Cucumberu je potřeba popsat proces, jak Cucumber funguje. Jeho grafická znázornění je prezentováno na obrázku (Obr. 12).

Po spuštění testů začne Cucumber číst z textových souborů, které se nazývají funkce (Features). Funkce obsahují scénáře (Scenarios) napsané v jazyce Gherkin. Syntaxe Gherkinu se musí dodržovat, aby Cucumber jednotlivým scénářům porozuměl. Tyto scénáře si načte a spustí je. Každý scénář je seznam kroků (Steps), které musí Cucumber zpracovat. Cucumber spolu s funkcemi dostane i sadu definic kroků (Step definitions), které mapují každý krok do kódu. Zde se provede akce, která je popsaná krokem. Akce představuje jeden a více řádků kódu (Support Code), které mohou využívat prvky z automatizační knihovny (Automation Library).

Pokud se kód v definici kroku provede správně, automaticky se přechází na další krok ve scénáři. Jestliže se Cucumber dostane až na konec scénáře a během jednotlivých kroků nezaznamená chybu, tak se scénář označí jako úspěšný. Pokud některý z kroků obsahuje chybu, Cucumber aktuální scénář označí za neúspěšný a přejde na další scénář. [27]



Obrázek 12 Cucumber architektura [27]

4.6.2 TestNG

Jedná se o testovací framework navržený ke zjednodušení široké škály testovacích potřeb od unit testů až po integrační testy. [28]

Spojení TestNG a Cucumberu umožní paralelní provádění testovacích scénářů. Takové zpracování testů umožní ušetřit značný čas při jejich průběhu.

II. PRAKTICKÁ ČÁST

5 METODIKA PRO TVORBU AUTOMATIZOVANÝCH TESTŮ PLATEBNÍCH TERMINÁLŮ NA PLATFORMĚ ANDROID

Při tvorbě metodiky pro automatizované testy platebních terminálů na platformě Android jsem vycházel z vlastních poznatků. Ty jsem získal během testování při vývoji platební aplikace na platebním terminálu Newland N910, který běží na platformě Android. Během manuálního testování a následně automatizovaného testování jsem se setkal s řadou problematik, které jsem musel překonat, aby bylo možné uvést automatizaci do provozu.

Metodiku jsem v základu rozdělil podle úrovní testování. Jedná se tedy o komponentní, integrační, systémové a akceptační úrovně. Pro jednotlivé úrovně jsem stanovil testovací sady, které považuji za základní doporučené sady v rámci testování aplikací pro platební terminály na platformě Android.

Testovací sady v metodice představují základní doporučené sady při testování aplikací pro platební terminály. Z testovacích sad lze při testování vycházet a následně si je rozšířit podle požadavků uživatele nebo funkcionalit, které jsou specifické pro vyvíjenou aplikaci.

Na základě vytvořené metodiky lze postupovat od samotného počátku testování aplikací platebních terminálů až do samotného konce, kdy dojde k oficiální certifikaci a schválení aplikace zákazníkem. Tím samotná cesta aplikace nekončí, protože požadavky zákazníků se mohou postupem času měnit a aplikaci je potřeba upravovat. Právě v takovém případě má automatizované testování aplikací platebních terminálů smysl.

5.1 Komponentní úroveň

O danou úroveň testů se ve většině případů starají sami vývojáři pomocí jednotkových testů. Jde o testování nejmenších částí platební aplikace na terminálech. Testování komponent probíhá již v rané fázi vývoje a dost často je tato část podceňovaná.

U komponentové úrovně se jedná o white-box testy. Aplikace pro platební terminály mají řadu komponent. Komponenty si můžeme rozdělit na dvě hlavní skupiny.

Jedna skupina se zabývá obsluhováním terminálu. Jedná se o komponenty, které vyvolávají zvukovou signalizaci, tiskárnu, světelnou signalizaci pomocí LED světel, čtení karet a další.

Druhá skupina komponent se zabývá zpracováním dat a řízením aplikace.

Problémy, které mohou vzniknout:

- Chyby v komponentách se mohou projevit až v pozdější fázi testování.
- Různé kombinace chyb v komponentách mohou vést k problematičtějšímu odhalení chyb a následně větší spotřebě času pro opravu.
- Čas pro odhalení a opravení chyb v pozdější fázi vývoje projektu stojí více finančních prostředků.
- Chyby u některých komponent se mohou projevit až v produkci, kdy je v oběhu už velké množství terminálů.

Jak předcházet zmíněným problémům:

- Stanovením si dostatečného minimálního procentuálního pokrytí jednotkovými testy u jednotlivých komponent.
- Při vývoji je potřeba jednotkové testy pravidelně používat pro kontrolu jednotlivých komponent, ve kterých se změny provedly.
- Udržovat jednotkové testy aktuální.
- Kontrolovat pokrytí komponent jednotkovými testy v projektu.

Cílem testování na komponentní úrovni, je dostatečně otestovat samostatnou funkčnost komponent, než se přejde do integrační úrovně. U aplikací, které manipulují s finančními prostředky zákazníka by pokrytí jednotkových testů mělo být v co nejvyšší možné míře, aby se zajistila očekávaná funkčnost jednotlivých komponent.

5.2 Integrační úroveň

Integrační úroveň se v některých případech spojuje se systémovou úrovní. U aplikací platebních terminálu je lepší raději integrační úroveň zachovat samostatnou a spojení se systémovou úrovní neprovádět. Při nedostatečném pokrytí testovacích případů v systémové úrovni by mohla uniknout chyba, na kterou by se dalo přijít při testování v integrační úrovni.

U testování integrační úrovně je potřeba přistupovat postupným přidáváním modulů. Po jejich řádném otestování lze přidávat další moduly, dokud se nespojí do výsledného systému. Po jejich spojení a otestování lze následně přejít na systémovou úroveň testování.

U aplikací platebních terminálů je potřeba ověřit správnou komunikaci mezi jednotlivými komponentami a komunikaci komponent s platebním terminálem.

Do integrační úrovně lze zahrnout testovací sady s black-box testy:

- Tisk účtenky
- Světelná a zvuková signalizace
- Platební karty
- GUI
- IP adresa, port, ID profilu
- Komunikace platebního terminálu
- Limity

5.2.1 Tisk účtenky

Účtenka je nedílnou součásti platebních terminálů, která slouží jako doklad pro zákazníka i pro obchodníka. V Čechách má účtenka ještě navíc charakter záručního dokladu zboží.

Kontrolu vytištěné účtenky nelze ověřit pomocí automatizovaných testů, pokud se nachází jen v papírové podobě. Naskýtá se zde více řešení.

Jedním řešením mohou být manuální testy, kdy se spoléhá na testera, který shlédne všechny vytištěné účtenky a porovná je. Takové řešení pro automatizaci není příliš vhodné.

Pokud se účtenka zobrazuje v elektronické podobě na obrazovce terminálu, může být kontrola účtenky snadnější. Takové řešení však není v aplikaci vždy přítomné.

Poslední z možností je úprava aplikace tak, aby před tiskem účtenky na papír ukládala obsah do souboru. Aplikace v takovém případě potřebuje oprávnění k uložení dat do Androidu, pokud již oprávnění nemá.

Problémy, které mohou vzniknout:

- Účtenka v elektronické formě obsahuje jiné údaje a formát než papírová účtenka.
- Na účtence chybí důležité údaje.
- Účtenka se nevytiskne.
- Tisk účtenky proběhne i bez papíru v terminálu. Bez chybové hlášky, který by na skutečnost upozorňovala a není možnost opětovného tisku účtenky po vložení nové ruličky papíru.

Jak předcházet zmíněným problémům:

- Díky automatizovaným testům je možnost kontroly obsahu účtenky. Tedy všechny potřebné údaje, které účtenka má obsahovat nebo ty, které jsou nastaveny na profilu pro daného uživatele.
- Kontrola grafické stránky účtenky na papíru se neobejde bez asistence testera.
- Potřeba testovat stavy, kdy dojde papír nebo kdy papír není vůbec vložený do terminálu.

5.2.2 Světelná a zvuková signalizace

Při různých operacích na platebních terminálech dochází ke zvukovým a světelným signalizacím. Většina signalizací se nejčastěji aktivuje při zpracování transakcí.

Světelná a zvuková signalizace nepředstavují kritickou funkčnost aplikace pro platební terminály. Pouze upozorňují klienta na skutečnost, která se na platebním terminálu odehrála. Přesto nelze testování signalizace opomenout. Testování signalizací lze provést několika způsoby. Automatizace je zde složitějším a dražším řešením.

První způsob je z pohledu manuálního testování, které se neshoduje s myšlenkou automatizace.

Druhý způsob testování již představuje automatizované testy. Pomocí rozhraní pro ovládání zvuku a světel, lze zjistit stav signalizace. Daný způsob testování lze provést pouze za předpokladu, že zmíněné rozhraní takovou funkcionalitu umožňuje.

Třetí možností je dovybavení externích zařízení, které budou monitorovat signalizace terminálu. Jedná se o nejdražší a nejkomplikovanější variantu. U signalizace, která nepředstavuje kritickou hrozbu z pohledu funkčnosti aplikace ji lze shledat zbytečnou.

Problémy, které mohou nastat:

- Zvuková ani světelná signalizace nebude aktivní.
- Spouštění signalizací nebude odpovídat navrženým situacím, což může klienta zmást.

Jak předcházet zmíněným problémům:

- Automatizovanými testy pomocí kontroly přes rozhraní pro signalizaci.
- Pomocí manuálních testů, kdy proběhne základní simulace situací, při kterých se signalizace používá.

5.2.3 Platební karty

V přechozích kapitolách nebyla platebním kartám věnována dostatečná pozornost. Existuje velké množství výrobců platebních karet. Každý druh platebních karet potřebuje vlastní kernel pro jejich zpracování. Výběr kernelu není závislý pouze na druhu platebních karet, ale i na jeho výrobci.

Vytvořit terminál, který bude schopný obsluhovat všechny druhy platebních karet ve většině případů není potřeba. Vše závisí na požadavcích obchodníka. Tím se liší i funkce platebního terminálu, tedy i obsah menu. Jiné platební karty bude přijímat síť benzinek nebo restaurace. Pro příklad je možné uvést stravenkové karty oproti běžným platebním kartám.

Problémy, které mohou vzniknout:

- Výběr špatného kernelu pro zvolený druh karet.
- Nefunkční vstup pro některý druh platebních karet.
- Nefunkční nastavené limity platebních karet pro dané vstupy.

Jak předcházet zmíněným problémům:

- Seznam testovaných karet by měl být sestaven podle schválených požadavků.
- Pro každou platební kartu by měly být provedeny testy všech možných vstupů. Jedná se o magnetický proužek, čip a bezkontaktní formu.
- Kontrola limitů platebních karet závislých na vstupu karty. Limity se dají nastavit podle požadavků.

5.2.4 GUI

Grafické uživatelské rozhraní tvoří vzhled aplikace, což u uživatele vyvolává první dojem, který bývá velmi důležitý. Poskládání vzhledu a rozmístění jednotlivých funkcionalit by pro uživatele mělo být snadné a intuitivní.

Kontrola GUI se neobejde bez manuálního zásahu. Zkušené oko testera snadně odhalí drobné změny v barvě, posunutí textu a dalších problematik, které jsou spojeny se vzhledem aplikace. Automatizovaná část dokáže kontrolovat viditelnost, možnost klikání, obsah elementů na jednotlivých obrazovkách aplikace.

Při testování GUI není potřeba sledovat dění uvnitř aplikace, důležitý je pouze konkrétní výsledek na konkrétní akci. Testy pro GUI se provádí pomocí black-box testů.

Problémy, které mohou nastat:

- Jednotlivé elementy se na obrazovkách nezobrazují.
- Na tlačítka nelze klikat.
- Obsah elementů není viditelný nebo není správný.
- Jednotlivé elementy na obrazovce jsou posunuté nebo špatně uspořádané.

Jak předcházet zmíněným problémům:

- Kontrolou jednotlivých obrazovek aplikace.
- Na obrazovkách je potřeba zkontrolovat viditelnost jednotlivých elementů a jejich obsah. U tlačítek je nutné zkontrolovat možnost klikání.

5.2.5 IP adresa, port, ID profilu

IP adresu, port i ID profilu jsou nezbytné pro možnost inicializace terminálu. Validaci vstupních hodnot lze řešit samostatně v testovací sadě pro validaci vstupních polí aplikace. Cílem testovací sady pro kontrolu IP adresy, portu a ID profilu, je ověření funkčnosti uložení zmíněných hodnot do aplikace.

Problémy, které mohou nastat:

- Neuložené nebo špatně uložené hodnoty vedou k chybnému směrování komunikace pro inicializaci terminálu.
- Přednastavené hodnoty z instalace terminálu mohou být chybné.

Jak předcházet zmíněným problémům:

- Testovat ukládané hodnoty v terminálu s jejich ověřením.
- Testovat ukládání chybných hodnot.
- Kontrolovat již předem vyplněné údaje z instalace terminálu.

5.2.6 Komunikace platebního terminálu

Platební aplikace na terminálu není uzavřený systém. Potřebuje komunikovat s platební bránou pro ověřování transakcí. Pokud zabezpečení komunikace nebude na dostatečné úrovni, útočník může pozměnit jednotlivé zprávy ve svůj prospěch.

Zpráva z terminálu, která obsahuje údaje zákazníka a obchodníka, může být změněna. Hrazená částka nemusí být připsána na obchodníkův bankovní účet, ale připíše se na účet útočníka.

Problémy, které mohou vzniknout:

- Použití komunikace, která je nebezpečná pro externí internetový provoz.
- Použití slabších a starších kryptografických algoritmů.
- Zasílání zpráv pouze v textovém formátu.
- Staré certifikáty, u kterých se neověřuje jejich platnost.
- Není vyžadovaná šifrovaná komunikace.

Jak předcházet zmíněným problémům:

- Kontrolou platnosti certifikátů.
- Zasíláním dat pomocí zabezpečených protokolů.
- Šifrováním dat pomocí silných kryptografických algoritmů.

Pro testování komunikace platebního terminálu lze využít například upraveného Smart hostu, který nám následně zasílá získaná data pro kontrolu. Tak lze otestovat obsah, ale i zabezpečení zpráv.

5.2.7 Limity

Limity pro platební aplikaci lze v našem případě nastavit přes profil. Pro limity lze nastavit minimální a maximální částku, ve které se platby budou pohybovat.

Problémy, které mohou nastat:

- Částky mimo minimální a maximální rozsah budou schváleny.
- Nezobrazení chybového hlášení v nekorektních situacích.
- Špatný formát chybových hlášení případně nekorektní zobrazené minimální a maximální nastavené částky.

Jak předcházet zmíněným problémům:

- Vytvořením testovacího profilu, který bude mít nastavenou minimální i maximální hranici limitu.
- Pomocí automatizačních testů, které budou testovat minimální, maximální, mezi maximem a minimem částku. Následně negativní scénáře s částkami mimo minimální a maximální hodnoty a kontrola chybových hlášení.

5.3 Systémová úroveň

Systémová úroveň je poslední úroveň testování před tím, než se aplikace začne certifikovat a předávat klientovi. Daná úroveň je pro nás z pohledu testování nejdůležitější. Jedná se o poslední možnost zachycení vážných nedostatků aplikace.

Systémová úroveň u aplikací pro platební terminály představuje testování, které často připomíná chování budoucích uživatelů. Pro představu lze uvést transakci prodej, kdy je nejdřív potřeba v terminálu zvolit akci prodeje, zadat částku, přiložit platební kartu, zadat pin a zkontrolovat stav transakce. Tento testovací případ postupně projde více vrstvami aplikace.

U systémové úrovně se implementují funkční a nefunkční testy. Nefunkční testy jsou v mnoha případech testovány samostatně. Je důležité, aby se na tyto testy nezapomínalo. Dále se v dané úrovni využívají testy typu black-box. Testování probíhá z pohledu zákazníka převážně přes grafické uživatelské rozhraní.

Do systémové úrovně můžeme zahrnout funkční testovací sady:

- Transakce prodej
- Transakce návrat
- Uzávěrka platebního terminálu
- Inicializace
- Prezentace citlivých dat
- Profily
- Ztráta dat
- Autentizace
- Aktualizace aplikace platebního terminálu

Nefunkční testovací sady:

- Performance

5.3.1 Transakce prodej

Jak už bylo zmíněno u systémové úrovně, transakce prodej patří mezi základní vlastnosti platebního terminálu. Bez funkčního prodeje se z terminálu stává krabička bez přidané hodnoty pro obchodníka. Zajištění bezchybné funkčnosti transakce prodeje je prioritou.

Problémy, které mohou nastat:

- V průběhu transakce nebude akceptovaný krok, kterým nám transakci zamítne.
- Dojde k chybě transakce, na kterou aplikace neumí zareagovat.
- Obrazovka pro zadání spropitného se nebude zobrazovat.
- Chybně zobrazené informace na obrazovce při průběhu transakce.
- Obrazovka pinu nebude pro určitý vstup nebo částku vyžadována.

Jak předcházet zmíněným problémům:

- Vytvořením testů, které budou kontrolovat transakce od zadání částky, přes pin, spropitné, průběh transakce až do jejich zdárného provedení. Tedy testovat prodej od samotného začátku až po úplný konec s ověřením záznamu v terminálu.
- Vytvořené testy je potřeba provést s různými kreditními kartami.
- V rámci transakce prodej by se měly řešit i rozdílné vstupy kreditních karet.
- Simulování neúspěšných stavů transakce prodej jako je přerušení internetového spojení, komunikace na špatnou IP adresu a další. Opět je u testů potřeba kontrolovat celý průběh a výsledek transakce.

5.3.2 Transakce návrat

Transakce návrat je od transakce prodej do jisté míry shodná, co se průběhu transakce týče. Problémy i způsoby předcházení problémů se ve většině oblastech shodují. Rozdíl je příkladem u obrazovky pro zadávání spropitného. Spropitné se u transakce návrat nezobrazuje.

Problémy, které mohou nastat:

- V průběhu transakce nebude akceptovaný krok, který nám transakci zamítne.
- Dojde k chybě transakce, na kterou aplikace neumí zareagovat.
- Chybně zobrazené informace na obrazovce při průběhu transakce.
- Obrazovka pinu nebude pro určitý vstup nebo částku vyžadována.

Jak předcházet zmíněným problémům:

- Vytvořením testů, které budou kontrolovat transakce od zadání částky, přes pin, průběh transakce až do jejich zdárného provedení. Tedy testovat návrat od samotného začátku až po úplný konec s ověřením záznamu v terminálu.
- Vytvořené testy je potřeba provést s různými kreditními kartami.

- V rámci transakce návrat by se měly řešit i rozdílné vstupy kreditních karet.
- Simulování neúspěšných stavů transakce návrat jako je přerušení internetového spojení, komunikace na špatnou IP adresu a další. Opět je u testů potřeba kontrolovat celý průběh a výsledek transakce.

5.3.3 Uzávěrka platebního terminálu

Uzávěrka se úzce spojuje s přehledem transakcí. Přehled transakcí obsahuje všechny vykonané transakce v průběhu dne. Po provedení uzávěrky se transakce z přehledu odstraní a výsledek se vytiskne.

Uzávěrku platebního terminálu lze využít, pokud je potřeba uzavřít pracovní den nebo zúčtovací období. Skvěle slouží ke kontrole kasy, pokud dochází ke změně obsluhy terminálu a je potřeba zkontrolovat celkovou částku finančních prostředků.

Problémy, které mohou nastat:

- Uzávěrka nebude odpovídat realitě. To může vést k chybě v účetnictví nebo problémovým situacím, pokud se u terminálu střídají směny.
- Uzávěrka nebude funkční. Pokud přehled transakcí není vyprázdněný, nelze provést inicializaci terminálu a tím pádem nahrát různé změny v nastavení terminálu. Následně je vyžadován servisní zásah, který dokáže přehled transakcí smazat. Uživatel tak přijde o data z uzávěrky.

Jak předcházet zmíněným problémům:

- Je potřeba si stanovit testy, které pokryjí stavy pro prázdnou uzávěrku, ale i pro uzávěrku, která obsahuje různé druhy transakcí.
- Důležité je zkontrolovat výsledek uzávěrky a kontrolní součty.
- Je nutné kontrolovat přehled transakcí, zda dochází k mazání transakcí po uzávěrce.

5.3.4 Inicializace

Inicializaci aplikace platebního terminálu lze považovat za základní operaci, která se prování hned po instalaci aplikace do platebního terminálu. Pomine-li se autentizace, nastavení portu, IP adresa, ID profil a SSL. Bez inicializace terminálu nedojde ke stažení profilu, který je určený pro konkrétního zákazníka a tím nedojde k základnímu zprovoznění terminálu.

Možnost provedení inicializace záleží na prvotním nastavení již zmíněných faktorů, jako je IP adresa, port, ID profilu a SSL. Inicializaci lze tedy testovat přímo samotnou nebo ji rozšířit i o změny ve zmíněných faktorech, kdy dojde k vyvolání různých stavů při inicializaci.

Problémy, které mohou nastat:

- Neúspěšná inicializace vede k nedokončení primárního nastavení terminálu pro zákazníka.
- Při aktualizaci platební aplikace se mohou přidávat nové komunikační tabulky do profilu uživatele. V takovém případě je potřeba provést inicializaci. Bez funkční inicializace nebude možné provádět transakce vlivem špatných komunikačních a certifikačních tabulek.

Jak předcházet zmíněným problémům:

- Stanovením a vytvořením automatizovaných testů pro kritické situace inicializace.
- Jsou potřeba pravidelné kontroly pro každou novou verzi aplikace.

5.3.5 Prezentace citlivých dat

Platební terminál pracuje s řadou citlivých dat zákazníků. Aby se tato data nedostala do cizích rukou, je potřeba je maskovat.

Problémy, které mohou vzniknout:

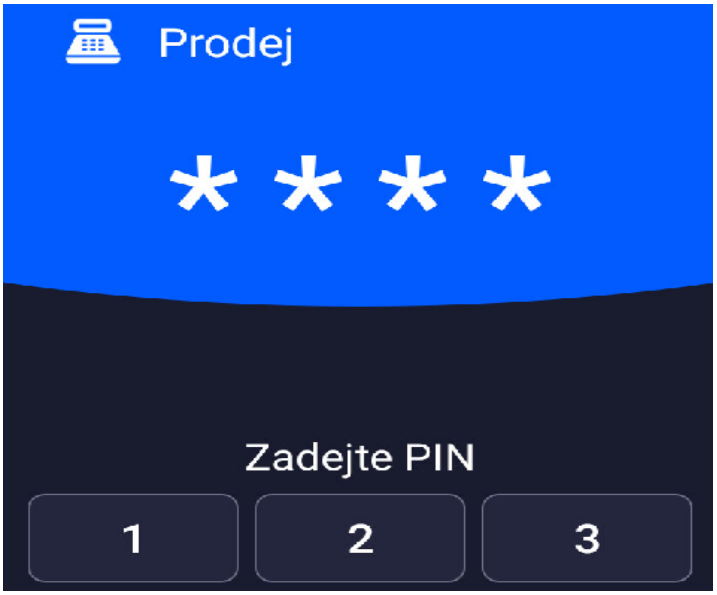
- Nemaskovaný PAN (primary account number) v přehledu vykonaných transakcí by vedl k prozrazení citlivého údaje o kreditní kartě zákazníka.
- Nemaskované zadávání PINu by mohlo znamenat další riziko pro zákazníka, při nepozornosti může být vyzrazen cizí osobě.
- Zobrazení citlivých údajů při průběhu transakce na obrazovce terminálu.
- Uložení originálního znění citlivých dat do databáze na terminálu.
- Tisk nesprávných údajů na papírovou účtenku. Účtenka může obsahovat špatný formát, neodpovídající obsah nebo nemaskovaný PAN.

Jak předcházet zmíněným problémům:

- Maskování citlivých údajů na obrazovce terminálu. Citlivé údaje by se neměly v originální formě ukládat do terminálu. Zahrnutím kontroly citlivých údajů do automatizovaných regresních testů.
- Kontrola databáze platebního terminálu po provedení transakce.

- Kontrola smazání dat z terminálu po provedení uzávěrky terminálu. Data by se v terminálu měla držet jen na potřebně dlouhou dobu.
- Vytisknutou účtenku je možné v automatizovaných testech jen stěží kontrolovat. Obsah účtenky lze pro testovací účely ukládat na platební terminál, kde jen na základě zvolené kreditní karty možné zkontrolovat všechny odpovídající údaje.

Obrázek 13 Maskovaný PAN na obrazovce platebního terminálu



Obrázek 14 Maskovaný PIN na obrazovce terminálu

5.3.6 Profily

Umožňují snadné přizpůsobení požadavků zákazníka. Každý zákazník má svůj unikátní profil. Pro zákazníka lze nastavit text na účtence, limity, komunikační tabulky, certifikace a další. Jedná se tedy o stěžejní prvek, který se stahuje při inicializaci aplikace pro platební terminál.

Problémy, které mohou nastat:

- Stažení profilu proběhne neúspěšně.
- Část z nastavení profilu se nepoužije v platební aplikaci. Vlivem nepoužitého nastavení z profilu může dojít k neúspěšnému provádění transakcí, tisku účtenek, požadovaným limitům při platbě, neshodujícím se certifikátům a další.

Jak předcházet zmíněným problémům:

- Testy při stahování inicializace kontrolují průběh stažení profilu i jeho načtení do aplikace.
- Pro různé druhy testů je potřeba mít nastavené odpovídající profily.

5.3.7 Ztráta dat

Jak už bylo zmíněno výše, terminál obsahuje řadu citlivých dat. Jejich ztráta by mohla obchodníkům přinést komplikace v podnikání. Stejný problém by mohl vzniknout u společnosti, která stojí za vývojem platební aplikaci na terminály. Tento problém by mohl způsobit velký počet servisních výjezdů a dalších komplikací, které by firmu mohly připravit o značný finanční obnos.

Problémy, které mohou vzniknout:

- Při pádu terminálu v době uzávěrky by obchodník mohl přijít o záznam transakcí, které v daný den byly provedeny.
- Při pádu aplikace dojde k vymazání nastavení terminálu, jako IP adresu, port, ID profilu uživatele a další. Opětovné nastavení terminálu by musel řešit servisní technik.
- Aplikace po standardním vypnutí si nepamatuje nastavené hodnoty nebo provedené transakce po opětovném spuštění.
- Po inicializaci terminálu dochází k chybnému nastavení dat.

Jak předcházet zmíněným problémům:

- Definovat stavy, při kterých by mohla být aplikace náchylnější na pády a podchytit je při vývoji.
- Simulovat stavy pádu aplikace a provádět následnou kontrolu nastavení terminálu i transakcí, které byly před pádem vytvořeny.

- Simulovat opakované zapnutí a vypnutí aplikace, sledovat nastavení terminálu, zda nedochází ke změně.
- Inicializovat terminál a zkontrolovat nastavené hodnoty terminálu.

5.3.8 Autentizace

Jedná se o důležitou část aplikací pro platební terminály. Chybná validace hesel může mít vážné následky pro řízení aplikace pro platební terminály.

Problémy, které mohou vzniknout:

- Neoprávněný přístup do zabezpečené části servisního menu aplikace platebního terminálu, kde jsou uloženy klíče terminálu a další důležité údaje.
- Nastavení slabého hesla nebo ponechání jednoduchého hesla, které bylo použito při vývoji aplikace do servisní části nebo k citlivým údajům.
- Po odhlášení se ze zabezpečené sekce lze opět dostat do zabezpečené sekce bez nutnosti opětovného zadávání přístupového hesla.
- Není nastavený čas odhlášení při nečinnosti v zabezpečené sekci terminálu.
- Nedostatečná validace hesel pro přístup do zabezpečené sekce.

Jak předcházet zmíněným problémům:

- Stanovit si pravidla pro vytváření dostatečně silných hesel.
- Testováním validace hesel pro odstranění případných nedostatků, které by mohly vést k ohrožení bezpečného fungování terminálu.
- Omezování možností neúspěšných pokusů o přihlášení se do zabezpečené sekce terminálu. Případně prodlužování času pro opětovný pokus.
- Kontrola automatického odhlášení.
- Kontrola odhlášení se ze zabezpečené sekce a následného pokusu o přistup do zabezpečené sekce bez opětovného zadání hesla.
- Zaznamenávání si větší četnost neúspěšných pokusů.

5.3.9 Validace vstupních polí aplikace

Aplikace na platebním terminálu mají řadu polí pro vkládání hodnot. Může se jednat o zadávání částky při běžné platbě, údaje pro nastavení terminálu, vkládání klíčů a další. V zájmu bezpečnosti a správné funkčnosti by validace měla probíhat pro všechny vstupy.

Problémy, které mohou vzniknout:

- Nastavení nereálné IP adresy, portu, částky pro zvolenou transakci nebo unikátní identifikátor uživatele.
- Možnost zadání libovolných znaků, které mohou vyvolat pád aplikace.
- Možnost vzniku útoku na aplikaci vlivem nevalidovaných polí.

Jak předcházet zmíněným problémům:

- Důkladně otestovat validaci všech polí v aplikaci.
- Omezit pole jen na potřebné znaky a maximální délku.

5.3.10 Aktualizace aplikace platebního terminálu

Aplikace v průběhu času potřebují aktualizace pro zvýšení bezpečnosti, přidání funkčností nebo opravu chyb. Pro aktualizaci platební aplikace terminálu může být aktualizace součástí platební aplikace nebo může pro tento účel sloužit samostatná aplikace, která je předinstalovaná při dodávce s terminálem.

Problémy, které mohou nastat:

- Pokud je aktualizace aplikace řešena pomocí druhé aplikace, hrozí zde nedostatečné zabezpečení pomocné aplikace. Při druhé aplikaci je potřeba testovat zmíněnou aplikaci samostatně.
- Pád při aktualizaci může ohrozit funkčnost platební aplikace na terminálu. Při spuštění platební aplikace se nenaváže na neúspěšnou aktualizaci.
- Chybná aktualizace znamená nefunkční část nebo celou platební aplikaci.

Jak předcházet zmíněným problémům:

- Využívat zabezpečené komunikační protokoly pro stahování aktualizace aplikace.
- Dostatečně testovat jednotlivé funkce aplikace.
- Otestovat aktualizace aplikace i při simulovaném pádu.

5.3.11 Performance

Platební aplikace na platebním terminálu může splňovat podmínky certifikace, ale neznamená to, že se automaticky stane oblíbenou u obchodníků. Performance aplikace na platebním terminálu je jeden z hlavních pilířů za cestou k úspěchu.

Pro představu lze zmínit případ, kdy je platební terminál denně zatížený velkým počtem transakcí.
Se vzrůstajícím počtem transakcí se rychlost terminálu postupně zpomaluje. Nakonec stav aplikace dojde do kritického stavu, kdy schválení a zpracování platby bude trvat několik minut. Obchodník po špatných zkušenostech vrátí terminál a nenávratně odchází ke konkurenci.

Problémy, které mohou vzniknout:

- Sekávání při přechodu mezi obrazovkami.
- U terminálů s integrovanou baterkou může dojít k nadměrné spotřebě energie.
- Dlouhá doba provedení jednotlivých transakcí.
- Zpomalování aplikace terminálu při vyšším počtu transakcí.
- Pád aplikace a v nejhorším případě ztráta dat.

Jak předcházet zmíněným problémům:

- Cílené generování vyššího počtu transakcí.
    - Sleduje se čas zpracování transakce. Transakce mohou představovat prodej, návrat, storno a nejvíc kritickou transakci uzávěrku, která se většinou provádí nakonec a dochází u ní ke zpracování všech vykonaných transakcí daného dne.
    - Sleduje se zatížení hardwaru terminálu.
    - Sleduje se stav baterie. Zde je potřeba zařídit vypnutí dobíjení terminálu přes kabel nebo testovat přes Wi-fi, může však dojít k částečnému zkreslení výsledku.
- Monkey testování je druh namáhavého testování, kterého lze dosáhnout náhodným i opakovaným způsobem. Uživatel si smí navolit frekvenci různých gest, které jsou generovány. Díky nim lze aplikaci dostat do kritických situací i následného pádu. Neočekávané chování je následně zaznamenáno.
- Vizuální kontrola mezi přechody obrazovek.
- Vývojáři by měli vytvářet aplikaci s ohledem na omezený výkon terminálu. Android poskytuje vývojářům řadu typů na psaní svižných aplikací.

Aby se předešlo zmíněných problému v produkci, je potřeba vždy na konci testovacího cyklu provést performance testování. Řadu performance problému lze odhalit i při počátečním manuálním testování nebo při vizuální kontrole aplikace.

5.4 Akceptační úroveň

Po úspěšném dokončení systémové úrovně již následuje akceptační úroveň, kde se řeší testovací scénáře, které se dohodly se zákazníkem. Tyto testovací scénáře mohou vycházet z testovacích scénářů, které se používají v automatizovaných testech pro platební terminály, ale není to vždy pravidlem.

Pro příklad lze uvést několik běžných testů pro akceptační úroveň, které se musí shodovat se stanovenými požadavky:

- Spadají zde jednotlivé transakce terminálu. Transakcí se týká i kontrola průběhu a výsledku transakce. Zda se na obrazovkách zobrazují stanovené informace.
- Přehled vykonaných transakcí a zobrazených informací.
- Kontrola funkcí aplikace platebního terminálu.
- Rychlost aplikace a její odolnost proti pádu nebo nečekaným stavům.
- Namátkové procházení terminálu, které má napodobovat běžného uživatele.
- Testovací provoz u uživatele.

6 DOPORUČENÉ KRITÉRIA PRO POUŽITÍ VYTVOŘENÉ METODIKY

Kritéria pro použití vytvořené metodiky se dělí na vstupní a výstupní. Dané kritéria nám zaručují, kdy a za jakých podmínek vytvořenou metodiku použít. V rámci těchto kritérií by metodika měla poskytovat očekáváné výsledky.

Jednotlivé kritéria lze upřesňovat podle požadované výsledné kvality aplikace. Před samotným testováním je potřeba si stanovit celkový počet a úroveň závažnosti chyb, které mohou být akceptované.

6.1 Vstupní kritéria

Vstupní kritéria určují, kdy lze vytvořenou metodiku uplatňovat při testování aplikace pro platební terminály. Nelze opomenout, že vytvořená metodika je navržena pro automatizované testování.

Základní vstupní kritéria pro vytvořenou metodiku:

- Testovací platební terminál s platformou Android
- Testovací platební karty
- Připravenost testovacího prostředí
- Připravenost testovacích dat
- Dostupná verze aplikace, které již může být testovaná
- Vývojový tým provedl všechny jednotkové testy a opravil objevené defekty

6.2 Výstupní kritéria

Výstupní kritéria určují, kdy testovací sady z metodiky splnily stanovený cíl nebo kdy lze testování prohlásit za dokončené. Použití vytvořené metodiky bez stanovených cílů přinese pouze zmatek do vývojového procesu aplikace.

Základní výstupní kritéria pro vytvořenou metodiku:

- Provedení všech plánovaných testovacích sad
- Prověření všech nalezených chyb
- Vyřešení všech závažných chyb
- Podle výsledků testů lze usoudit, že bylo dosaženo stabilního a spolehlivého produktu

7 OBECNÉ TESTOVACÍ SADY

Na základě vytvořené metodiky, bylo vybráno několik testovacích sad. Zvolené sady byly rozšířené o testovací případy. Testovacími případy lze přímo uplatnit pro automatizované testování aplikací pro platební terminály. Očekávané výsledky případů se mohou mírně lišit. Vždy záleží na konkrétní implementaci aplikace.

Označení jednotlivých testovacích sad a testovacích případů je pomocí unikátních ID. Použité identifikátory jsou pouze ukázkové. V běžné praxi se používá pouze číselné označení bez písmen.

Pro větší přehlednost byl přidán strom vazeb s testovacími sadami a testovacími případy, které jsou obsažené v aktuální kapitole, viz příloha P II.

7.1 Testovací sada pro autentizaci v platebním terminálu

ID = 1A

Popis: Cílem sady je testovat všechny autentizace, které v aplikacích pro platební terminály mohou nastat. V uvedených testovacích případech se jedná o přístup do servisního menu, do kterého běžný uživatel nemá oprávnění přistupovat. Daná část je přístupná pouze pro servisní techniky.

Testovací případy se nezabývají pouze autentizací, ale i automatickým odhlášením uživatele.

Podmínky: Aplikace bude ve stavu, ve kterém se platební terminál s aplikací dodává zákazníkovi. Prvotní nastavení aplikace je již provedeno.

7.1.1 Kontrola validní autentizace do servisního menu

ID = 1A001
Typ testu: Komponentní
Kroky:

1. Jděte na obrazovku pro přihlášení do servisního menu
2. Zadejte správné přihlašovací heslo
3. Potvrďte přihlašovací heslo
4. Zkontrolujte přihlášení

Očekávaný výsledek:

- Heslo bude úspěšně validováno.
- Uživatel bude přihlášený do servisního menu.

7.1.2 Kontrola autentizace bez zadání hesla do servisního menu

ID = 1A002
Typ testu: Komponentní
Kroky:
1. Jděte na obrazovku pro přihlášení do servisního menu
2. Potvrďte přihlašovací heslo
3. Zkontrolujte neúspěšné přihlášení

Očekávaný výsledek:
- Heslo nebude přijato.
- Objeví se chybová hláška pro špatně zadané heslo.

7.1.3 Kontrola autentizace se zadáním hesla o minimální délce počtu znaků do servisního menu

ID = 1A003
Typ testu: Komponentní
Kroky:
1. Jděte na obrazovku pro přihlášení do servisního menu
2. Zadejte heslo o minimální délce znaků
3. Potvrďte přihlašovací heslo
4. Zkontrolujte neúspěšné přihlášení

Očekávaný výsledek:
- Heslo nebude přijato.
- Objeví se chybová hláška pro špatně zadané heslo.

7.1.4 Kontrola autentizace se zadáním hesla o maximální délce počtu znaků do servisního menu

ID = 1A004
Typ testu: Komponentní
Kroky:
1. Jděte na obrazovku pro přihlášení do servisního menu
2. Zadejte heslo o maximální délce znaků
3. Potvrďte přihlašovací heslo
4. Zkontrolujte neúspěšné přihlášení

Očekávaný výsledek:
- Heslo nebude přijato.
- Objeví se chybová hláška pro špatně zadané heslo.

7.1.5 Kontrola autentizace se zadáním hesla o jedno delší, než je maximum do servisního menu

ID = 1A005

Typ testu: Komponentní
Kroky:

1. Jděte na obrazovku pro přihlášení do servisního menu
2. Zadejte heslo o znak větší, než je maximální délka
3. Zkontrolujte počet zobrazených zadaných znaků na obrazovce
4. Potvrďte přihlašovací heslo
5. Zkontrolujte neúspěšné přihlášení

Očekávaný výsledek:

- Heslo nebude přijato.
- Objeví se chybová hláška pro špatně zadané heslo.
- Počet zobrazených znaků na obrazovce bude o jeden menší, než je zadaná hodnota.

7.1.6 Kontrola autentizace se zadáním nevalidních znaků pro přihlášení do servisního menu

ID = 1A006
Typ testu: Komponentní
Kroky:

1. Jděte na obrazovku pro přihlášení do servisního menu
2. Zadejte heslo s nevalidními znaky
3. Zkontrolujte počet zobrazených znaků na obrazovce
4. Potvrďte přihlašovací heslo
5. Zkontrolujte neúspěšné přihlášení

Očekávaný výsledek:

- Heslo nebude přijato.
- Objeví se chybová hláška pro špatně zadané heslo.
- Počet zobrazených znaků na obrazovce bude 0

7.1.7 Kontrola maskování zadaných čísel hesla do servisního menu

ID = 1A007
Typ testu: Komponentní
Kroky:

1. Jděte na obrazovku pro přihlášení do servisního menu
2. Zadejte správné přihlašovací heslo
3. Zkontrolujte délku zobrazených znaků a jejich formu

Očekávaný výsledek:

- Počet zobrazených znaků bude odpovídat počtu zadaných znaků
- Zadaná znaky jsou na obrazovce maskovány

7.1.8 Odhlášení uživatele ze servisního menu

ID = 1A008
Typ testu: Komponentní
Kroky:

1. Jděte na obrazovku pro přihlášení do servisního menu
2. Zadejte správné přihlašovací heslo
3. Potvrďte přihlašovací heslo
4. Odhlaste se ze servisního menu
5. Zkontrolujte odhlášení uživatele
6. Jděte na obrazovku pro přihlášení do servisního menu
7. Zkontrolujte přihlašovací obrazovku pro zadání hesla

Očekávaný výsledek:

- Po odhlášení, se uživatel dostane na odpovídající obrazovku
- Po odhlášení je vyžadováno opětovné zadání hesla pro vstup do servisního menu

7.1.9 Opakované neúspěšné zadávání hesla do servisního menu

ID = 1A009
Typ testu: Komponentní
Kroky:

1. Jděte na obrazovku pro přihlášení do servisního menu
2. Zadejte chybné přihlašovací heslo
3. Potvrďte přihlašovací heslo
4. Zkontrolujte neúspěšné přihlášení
5. Zadejte chybné přihlašovací heslo
6. Potvrďte přihlašovací heslo
7. Zkontrolujte neúspěšné přihlášení
8. Zadejte chybné přihlašovací heslo
9. Potvrďte přihlašovací heslo
10. Zkontrolujte neúspěšné přihlášení a chybové hlášení pro přesáhnutí počtu neúspěšných autentizací

Očekávaný výsledek:

- Zobrazení chybové hlášky pro maximální počet neúspěšných pokusů autentizace

7.1.10 Odhlášení uživatele ze servisního menu v případě nečinnosti 30 sekund

ID = 1A010
Typ testu: Komponentní
Kroky:

1. Jděte na obrazovku pro přihlášení do servisního menu
2. Zadejte správné přihlašovací heslo
3. Potvrďte přihlašovací heslo
4. Počkejte 30 sekund na automatické odhlášení
5. Zkontrolujte odhlášení uživatele a návrat na odpovídající obrazovku
6. Jděte na obrazovku pro přihlášení do servisního menu
7. Zkontrolujte přihlašovací obrazovku

Očekávaný výsledek:

- Po 30 sekundách dojde k automatickému odhlášení
- Po odhlášení se uživatel dostane na odpovídající obrazovku
- Při opětovném přístupu do servisního menu je potřeba znovu zadat heslo

7.2 Testovací sada pro validaci vstupních polí aplikace

ID = 2A

Popis: Testovací sada obsahující testovací případy pro otestování vstupních polí aplikace. Pro ukázku jsou zde vytvořeny testy zadávání hodnot transakcí, portu, ID profilu a další.

Podmínky: Aplikace je již po prvotním nastavení a inicializaci terminálu.

7.2.1 Zadání validní celočíselné částky pro transakci prodej

ID = 2A001
Typ testu: Systémový
Kroky:
1. Na obrazovce pro transakci prodej zadejte validní celočíselnou částku
2. Potvrďte zadanou částku
3. Zkontrolujte schválení zadané částky

Očekávaný výsledek:
- Uživateli by se měla zobrazit obrazovka s výzvou k uhrazení částky
- Zadaná částka je schválena a odpovídá zadané částce na obrazovce

7.2.2 Zadání validní částky s desetinou čárkou pro transakci prodej

ID = 2A002
Typ testu: Systémový
Kroky:
1. Na obrazovce pro transakci prodej zadejte validní částku s desetinou čárkou
2. Potvrďte zadanou částku
3. Zkontrolujte schválení zadané částky

Očekávaný výsledek:
- Uživateli by se měla zobrazit obrazovka s výzvou k uhrazení částky
- Zadaná částka je schválena a odpovídá zadané částce na obrazovce

7.2.3 Zadání nulové částky pro transakci prodej

ID = 2A003
Typ testu: Systémový
Kroky:
1. Na obrazovce pro transakci prodej zadejte nulovou částku
2. Potvrďte zadanou částku
3. Zkontrolujte obrazovku pro transakci prodej

Očekávaný výsledek:
- Nulová částka nevstoupí do procesu prodeje, ale zůstane na hlavní obrazovce beze změn

7.2.4 Validace celočíselné částky o jedna větší, než je maximum pro transakci prodej

ID = 2A004
Typ testu: Systémový
Kroky:
1. Na obrazovce pro transakci prodej zadejte o jedna větší částku, než je maximum
2. Zkontrolujte z validovanou částku zobrazenou na obrazovce

Očekávaný výsledek:
- Maximální délka celočíselné částky je 1 pozici menší než zadaná částka.

7.2.5 Validace částky s několika desetinnými čárky, celočíselnou hodnotou vyšší, než je maximum a 3 desetinnými pozicemi pro transakci prodej

ID = 2A005
Typ testu: Systémový
Kroky:
1. Na obrazovce pro transakci prodej zadejte nevalidní částku
2. Zkontrolujte z validovanou částku

Očekávaný výsledek:
- Výsledkem validace bude validní částka nebo chybová hláška

7.2.6 Validace vložení nečíselného řetězce pro transakci prodej

ID = 2A006
Typ testu: Systémový
Kroky:
1. Na obrazovce pro transakci prodej zadejte nečíselný řetězec
2. Zkontrolujte z validovaný řetězec

Očekávaný výsledek:
- Zobrazená z validovaná hodnota bude 0

7.2.7 Zadání validní celočíselné částky pro transakci návrat

ID = 2A007
Typ testu: Systémový
Kroky:
1. Na obrazovce pro transakci návrat zadejte validní celočíselnou částku
2. Potvrďte zadanou částku
3. Zkontrolujte schválení zadané částky

Očekávaný výsledek:
- Uživateli by se měla zobrazit obrazovka s výzvou k uhrazení částky
- Zadaná částka je schválena a odpovídá zadané částce na obrazovce

7.2.8 Zadání validní částky s desetinou čárkou pro transakci návrat

ID = 2A008
Typ testu: Systémový
Kroky:
1. Na obrazovce pro transakci návrat zadejte validní částku s desetinou čárkou
2. Potvrďte zadanou částku
3. Zkontrolujte schválení zadané částky

Očekávaný výsledek:
- Uživateli by se měla zobrazit obrazovka s výzvou k uhrazení částky
- Zadaná částka je schválena a odpovídá zadané částce na obrazovce

7.2.9 Zadání nulové částky pro transakci návrat

ID = 2A009
Typ testu: Systémový
Kroky:
1. Na obrazovce pro transakci návrat zadejte nulovou částku
2. Potvrďte zadanou částku
3. Zkontrolujte obrazovku pro transakci návrat

Očekávaný výsledek:
- Nulová částka nevyvolá akci. Uživatel nadále zůstane na obrazovce pro transakci návrat

7.2.10 Validace celočíselné částky o jedna větší, než je maximum pro transakci návrat

ID = 2A010
Typ testu: Systémový
Kroky:
1. Na obrazovce pro transakci návrat zadejte o jedna větší částku, než je maximum
2. Zkontrolujte z validovanou částku zobrazenou na obrazovce

Očekávaný výsledek:
- Maximální délka celočíselné částky je 1 pozici menší než zadaná částka.

7.2.11 Validace částky s několika desetinnými čárky, celočíselnou hodnotou vyšší, než je maximum a 3 desetinnými pozicemi pro transakci návrat

ID = 2A011
Typ testu: Systémový
Kroky:
1. Na obrazovce pro transakci návrat zadejte nevalidní částku
2. Zkontrolujte z validovanou částku

Očekávaný výsledek:
- Výsledkem validace bude validní částka nebo chybová hláška

7.2.12 Validace vložení nečíselného řetězce pro transakci návrat

ID = 2A012
Typ testu: Systémový
Kroky:
1. Na obrazovce pro transakci návrat zadejte nečíselný řetězec
2. Zkontrolujte z validovaný řetězec

Očekávaný výsledek:
Zobrazená z validovaná hodnota bude 0

7.2.13 Validace existujícího portu

ID = 2A013
Typ testu: Systémový
Kroky:
1. Jděte na obrazovku pro zadání portu
2. Zadejte hodnotu existujícího portu
3. Validace zadaného portu proběhne úspěšně

Očekávaný výsledek:
- Z validovaná hodnota portu bude shodná se zadanou hodnotou
- Validace neomezí zadávanou hodnotu portu

7.2.14 Validace nejnižšího čísla portu

ID = 2A014
Typ testu: Systémový
Kroky:
1. Jděte na obrazovku pro zadání portu
2. Zadejte nejnižší hodnotu portu
3. Validace zadaného portu proběhne úspěšně

Očekávaný výsledek:
- Z validovaná hodnota portu bude shodná se zadanou hodnotou
- Validace neomezí zadávanou hodnotu portu

7.2.15 Validace nevyššího čísla portu

ID = 2A015
Typ testu: Systémový
Kroky:
1. Jděte na obrazovku pro zadání portu
2. Zadejte nejvyšší hodnotu portu
3. Validace zadaného portu proběhne úspěšně

Očekávaný výsledek:
- Z validovaná hodnota portu bude shodná se zadanou hodnotou
- Validace neomezí zadávanou hodnotu portu

7.2.16 Validace o jedno vyšší číslo, než je maximum čísla portu

ID = 2A016
Typ testu: Systémový
Kroky:
1. Jděte na obrazovku pro zadání portu
2. Zadejte hodnotu portu o jednu číslo vyšší
3. Zkontrolujte z validovanou hodnotu portu

Očekávaný výsledek:
- Z validované číslo portu bude čtyřmístné nebo se zahlásí chybová hláška

7.2.17 Validace čísla portu s hodnotou 999999

ID = 2A017
Typ testu: Systémový
Kroky:
1. Jděte na obrazovku pro zadání portu
2. Zadejte hodnotu portu s číslem 999999
3. Zkontrolujte z validovanou hodnotu portu

Očekávaný výsledek:
- Číslo portu bude čtyřmístné nebo se zahlásí chybová hláška

7.2.18 Validace nečíselného řetězce portu

ID = 2A018
Typ testu: Systémový
Kroky:
1. Jděte na obrazovku pro zadání portu
2. Zadejte nečíselnou hodnotu portu
3. Zkontrolujte z validovanou hodnotu portu

Očekávaný výsledek:
- Číslo portu bude prázdné

7.2.19 Validace validního ID profilu

ID = 2A019
Typ testu: Systémový
Kroky:
1. Jděte na obrazovku pro zadání ID profilu
2. Zadejte validní ID profilu
3. Zkontrolujte z validovanou hodnotu ID profilu

Očekávaný výsledek:
- ID profilu bude úspěšně validováno
- Z validovaný výsledek bude shodný se zadávanou hodnotou

7.2.20 Validace ID profilu s větší délkou ID, než je povoleno

ID = 2A020
Typ testu: Systémový
Kroky:
1. Jděte na obrazovku pro zadání ID profilu
2. Zadejte ID profilu o délce 1 větší, než je povoleno
3. Zkontrolujte z validovanou hodnotu ID profilu

Očekávaný výsledek:
- Zobrazené ID profilu bude o maximální délce, která je povolena

7.2.21 Validace správné IP adresy

ID = 2A021
Typ testu: Systémový
Kroky:
1. Jděte na obrazovku pro zadání IP adresy
2. Zadejte validní IP adresu
3. Zkontrolujte z validovanou IP adresu

Očekávaný výsledek:
- IP adresa bude úspěšně validovaná a zobrazená na obrazovce

7.2.22 Validace nevalidní IP adresy

ID = 2A022
Typ testu: Systémový
Kroky:
1. Jděte na obrazovku pro zadání IP adresy
2. Zadejte nevalidní IP adresu
3. Zkontrolujte z validovanou IP adresu na obrazovce

Očekávaný výsledek:
- IP adresa bude úspěšně validovaná
- Nevalidní znaky budou z IP adresy odstraněny

7.2.23 Validace textového řetězce místo IP adresy

ID = 2A023
Typ testu: Systémový
Kroky:
1. Jděte na obrazovku pro zadání IP adresy
2. Zadejte textový řetězec místo IP adresy
3. Zkontrolujte z validovanou IP adresu na obrazovce

Očekávaný výsledek:
- Vložený textový řetězec bude validovaný
- Prázdná IP adresa na obrazovce pro nastavení IP adresy

7.3 Testovací sada pro kontrolu obrazovek

ID = 3A

Popis: Testovací sada obsahující testovací případy pro kontrolu zobrazených elementů na jednotlivých obrazovkách aplikace. Testy pro kontrolu obrazovek se v různých aplikacích mohou lišit. Opět záleží na návrhu aplikace a rozmístění jednotlivých funkcionalit v rámci ní.

Podmínky: Aplikace je již po prvotním nastavení a inicializaci terminálu.

7.3.1 Kontrola elementů na hlavní obrazovce

ID = 3A001
Typ testu: Komponentní
Kroky:
1. Jděte na hlavní obrazovku aplikace
2. Zkontrolujte viditelné elementy
3. Zkontrolujte text viditelných elementů

Očekávaný výsledek:
- Elementy a texty pro hlavní obrazovku jsou viditelné

7.3.2 Kontrola elementů na bočním menu

ID = 3A002
Typ testu: Komponentní
Kroky:
1. Jděte na obrazovku boční menu
2. Zkontrolujte viditelné elementy
3. Zkontrolujte text viditelných elementů

Očekávaný výsledek:
- Elementy a texty pro boční menu jsou viditelné

7.3.3 Kontrola elementů na transakčním menu

ID = 3A003
Typ testu: Komponentní
Kroky:
1. Jděte na obrazovku transakční menu
2. Zkontrolujte viditelné elementy
3. Zkontrolujte texty viditelných elementů

Očekávaný výsledek:
- Elementy a texty pro transakční menu jsou viditelné

7.3.4 Kontrola elementů pro transakci prodej

ID = 3A004
Typ testu: Komponentní
Kroky:
1. Jděte na obrazovku transakce prodej
2. Zkontrolujte viditelné elementy
3. Zkontrolujte texty viditelných elementů

Očekávaný výsledek:
- Elementy a texty na obrazovce pro prodej jsou viditelné

7.3.5 Kontrola elementů pro transakci návrat

ID = 3A005
Typ testu: Komponentní
Kroky:
1. Jděte na obrazovku transakce návrat
2. Zkontrolujte viditelné elementy
3. Zkontrolujte testy viditelných elementů

Očekávaný výsledek:
- Elementy a texty na obrazovce pro návrat jsou viditelné

7.3.6 Kontrola elementů pro menu dávky

ID = 3A006
Typ testu: Komponentní
Kroky:
1. Jděte na obrazovku menu dávky
2. Zkontrolujte viditelné elementy
3. Zkontrolujte texty viditelných elementů

Očekávaný výsledek:
- Elementy a texty na obrazovce pro menu dávky jsou viditelné

7.3.7 Kontrola elementů pro menu nastavení

ID = 3A007
Typ testu: Komponentní
Kroky:
1. Jděte na obrazovku menu nastavení
2. Zkontrolujte viditelné elementy
3. Zkontrolujte texty viditelných elementů

Očekávaný výsledek:
- Elementy a texty pro menu nastavení jsou viditelné

7.3.8 Kontrola elementů nastavení jazyku

ID = 3A008
Typ testu: Komponentní
Kroky:
1. Jděte na obrazovku nastavení jazyka
2. Zkontrolujte viditelné elementy
3. Zkontrolujte texty viditelných elementů

Očekávaný výsledek:
- Elementy a texty na obrazovce pro výběr jazyku jsou viditelné

7.3.9 Kontrola elementů na přihlašovací obrazovce do servisního menu

ID = 3A009
Typ testu: Komponentní
Kroky:
1. Jděte na obrazovku přihlášení do servisního menu
2. Zkontrolujte viditelné elementy
3. Zkontrolujte texty viditelných elementů

Očekávaný výsledek:
- Elementy a texty na přihlašovací obrazovce do servisního menu jsou viditelné

7.3.10 Kontrola elementů servisního menu

ID = 3A010
Typ testu: Komponentní
Kroky:
1. Jděte na obrazovku servisní menu
2. Zkontrolujte viditelné elementy
3. Zkontrolujte texty viditelných elementů

Očekávaný výsledek:
- Elementy a texty pro servisní menu jsou viditelné

7.3.11 Kontrola elementů správy klíčů

ID = 3A011
Typ testu: Komponentní
Kroky:
1. Jděte na obrazovku správy klíčů
2. Zkontrolujte viditelné elementy
3. Zkontrolujte texty viditelných elementů

Očekávaný výsledek:
- Elementy a texty pro správu klíčů jsou viditelné

7.3.12 Kontrola elementů inicializačního menu

ID = 3A012
Typ testu: Komponentní
Kroky:
1. Jděte na obrazovku inicializační menu
2. Zkontrolujte viditelné elementy
3. Zkontrolujte texty viditelných elementů

Očekávaný výsledek:
- Elementy a texty pro inicializační menu jsou viditelné

7.3.13 Kontrola elementů pro menu nastavení inicializace

ID = 3A013
Typ testu: Komponentní
Kroky:
1. Jděte na obrazovku nastavení inicializace
2. Zkontrolujte viditelné elementy
3. Zkontrolujte texty viditelných elementů

Očekávaný výsledek:
- Elementy a texty pro menu nastavení inicializace jsou viditelné

7.3.14 Kontrola elementů nastavení portu

ID = 3A014
Typ testu: Komponentní
Kroky:
1. Jděte na obrazovku nastavení portu
2. Zkontrolujte viditelné elementy
3. Zkontrolujte texty viditelných elementů

Očekávaný výsledek:
- Elementy a texty pro obrazovku nastavení portu jsou viditelné

7.3.15 Kontrola elementů nastavení IP adresy

ID = 3A015
Typ testu: Komponentní
Kroky:
1. Jděte na obrazovku nastavení IP adresy
2. Zkontrolujte viditelné elementy
3. Zkontrolujte texty viditelných elementů

Očekávaný výsledek:
- Elementy a texty pro obrazovku nastavení IP adresy jsou viditelné

7.3.16 Kontrola elementů ID profilu

ID = 3A016
Typ testu: Komponentní
Kroky:
1. Jděte na obrazovku ID profilu
2. Zkontrolujte viditelné elementy
3. Zkontrolujte texty viditelných elementů

Očekávaný výsledek:
- Elementy a texty pro obrazovku nastavení ID profilu jsou viditelné

7.3.17 Kontrola elementů nastavení SSL

ID = 3A017
Typ testu: Komponentní
Kroky:
1. Jděte na obrazovku nastavení SSL
2. Zkontrolujte viditelné elementy
3. Zkontrolujte texty viditelných elementů

Očekávaný výsledek:
- Elementy a texty pro obrazovku nastavení SSL jsou viditelné

7.3.18 Kontrola elementů informace terminálu

ID = 3A018
Typ testu: Komponentní
Kroky:
1. Jděte na obrazovku informace terminálu
2. Zkontrolujte viditelné elementy
3. Zkontrolujte texty viditelných elementů

Očekávaný výsledek:
- Elementy a texty pro obrazovku informace terminálu jsou viditelné

7.4 Testovací sada pro inicializaci terminálu

ID = 4A

Popis: Testovací sada obsahuje testovací scénáře pro kontrolu funkčnosti inicializace terminálu a částí aplikace, které s inicializací terminálu souvisí.

Podmínky: Aplikace je již po prvotním nastavení a inicializaci terminálu. Pro správnou funkčnost je ještě potřeba připojení terminálu k internetu.

7.4.1 Úspěšné provedení inicializace ze servisního menu

ID = 4A001
Typ testu: Systémový

Kroky:
1. Přihlaste se do servisního menu
2. Jděte na obrazovku inicializační menu
3. Proveďte inicializaci
4. Zkontrolujte výsledek inicializace

Očekávaný výsledek:
- Inicializace proběhne úspěšně

7.4.2 Úspěšné provedení inicializace z uživatelského nastavení

ID = 4A002
Typ testu: Systémový
Kroky:
1. Jděte na obrazovku nastavení
2. Proveďte inicializaci
3. Zkontrolujte stav inicializace

Očekávaný výsledek:
- Inicializace proběhne úspěšně

7.4.3 Inicializace s neexistujícím ID profilu

ID = 4A003
Typ testu: Systémový
Kroky:
1. Jděte na obrazovku nastavení ID profilu
2. Zadejte neexistující ID profilu
3. Potvrďte zadané ID profilu
4. Jděte na obrazovku inicializační menu
5. Proveďte inicializaci
6. Zkontrolujte stav inicializace

Očekávaný výsledek:
- Inicializace bude neúspěšná
- Objeví se chybová hláška o neúspěšné inicializaci

7.4.4 Inicializace se špatnou IP adresou

ID = 4A004
Typ testu: Systémový
Kroky:
1. Jděte na obrazovku nastavení IP adresy
2. Zadejte neexistují IP adresu
3. Potvrďte zadanou IP adresu
4. Jděte na obrazovku inicializační menu
5. Proveďte inicializaci
6. Zkontrolujte stav inicializace

Očekávaný výsledek:
- Inicializace bude neúspěšná

- Objeví se chybová hláška o neúspěšné inicializaci

7.4.5 Inicializace se špatným portem

ID = 4A005
Typ testu: Systémový
Kroky:
1. Jděte na obrazovku nastavení portu
2. Zadejte neexistují port
3. Potvrďte zadaný port
4. Jděte na obrazovku inicializační menu
5. Potvrďte inicializaci
6. Zkontrolujte stav inicializace

Očekávaný výsledek:
- Inicializace bude neúspěšná
- Objeví se chybová hláška o neúspěšné inicializaci

7.4.6 Inicializace se zakázaným SSL

ID = 4A006
Typ testu: Systémový
Kroky:
1. Jděte na obrazovku nastavení SSL
2. Proveďte zakázání SSL
3. Jděte na obrazovku inicializační menu
4. Proveďte inicializaci
5. Zkontrolujte stav inicializace

Očekávaný výsledek:
- Inicializace bude neúspěšná
- Objeví se chybová hláška o neúspěšné inicializaci

7.4.7 Inicializace bez připojení k internetu

ID = 4A007
Typ testu: Systémový
Kroky:
1. Vypněte Wi-fi připojení
2. Jděte na obrazovku inicializační menu
3. Proveďte inicializaci
4. Zkontrolujte stav inicializace
5. Zapněte Wi-fi připojení

Očekávaný výsledek:
- Inicializace bude neúspěšná
- Chybová hláška oznamující chybu spojení

7.5 Testovací sada pro nastavení IP adresy, portu, ID profilu, SSL

ID = 5A

Popis: Testovací sada obsahuje testovací případy, které testují nastavování a ukládání IP adresy, portu, ID profilu a SSL. Validace vstupních polí se řeší v samostatné testovací sadě.

Podmínky: Aplikace je již po prvotním nastavení a inicializaci terminálu. Nastavení obsahuje základní IP adresu, port, ID profilu a povolení SSL.

7.5.1 Kontrola povolení SSL

ID = 5A001
Typ testu: Komponentní
Kroky:
1. Jděte na obrazovku nastavení SSL
2. Proveďte povolení SSL
3. Opětovně jděte na obrazovku nastavení SSL
4. Zkontrolujte nastavenou hodnotu SSL

Očekávaný výsledek:
- SSL bude povoleno

7.5.2 Kontrola zakázaní SSL

ID = 5A002
Typ testu: Komponentní
Kroky:
1. Jděte na obrazovku nastavení SSL
2. Proveďte zakázání SSL
3. Opětovně jděte na obrazovku nastavení SSL
4. Zkontrolujte nastavenou hodnotu SSL

Očekávaný výsledek:
- SSL bude zakázáno

7.5.3 Uložení validního ID profilu

ID = 5A003
Typ testu: Komponentní
Kroky:
1. Jděte na obrazovku nastavení ID profilu
2. Zadejte validní ID profilu
3. Proveďte potvrzení zadané hodnoty ID profilu
4. Opětovně jděte na obrazovku nastavení ID profilu
5. Zkontroluj hodnotu ID profilu

Očekávaný výsledek:
- ID profilu je úspěšně uloženo a zobrazeno na obrazovce

7.5.4 Uložení ID profilu s prázdným ID

ID = 5A004
Typ testu: Komponentní
Kroky:
1. Jděte na obrazovku nastavení ID profilu
2. Zadejte prázdné ID profilu
3. Proveďte potvrzení zadané hodnoty ID profilu
4. Zkontrolujte chybovou hlášku o nesprávné hodnotě ID profilu

Očekávaný výsledek:
- Chybová hláška o nevalidní hodnotě ID profilu
- Neuložení chybného ID profilu

7.5.5 Kontrola uložení nesprávné délky ID profilu

ID = 5A005
Typ testu: Komponentní
Kroky:
1. Jděte na obrazovku nastavení ID profilu
2. Zadejte nevalidní ID profilu
3. Proveďte potvrzení zadané hodnoty ID profilu
4. Zkontrolujte chybovou hlášku o nesprávné hodnotě ID profilu

Očekávaný výsledek:
- Chybová hláška o nevalidní hodnotě ID profilu
- Neuložení chybného ID profilu

7.5.6 Kontrola uloženého ID profilu, odchodu bez změn a následného příchodu na obrazovku s ID profilem a provedení jeho kontroly

ID = 5A006
Typ testu: Komponentní
Kroky:
1. Jděte na obrazovku nastavení ID profilu
2. Proveďte kontrolu hodnoty ID profilu
3. Odejděte z obrazovky nastavení ID profilu beze změn
4. Opětovně jděte na obrazovku nastavení ID profilu
5. Zkontrolujte na obrazovce zobrazené ID profilu

Očekávaný výsledek:
- Uložená hodnota ID profilu bude beze změny

7.5.7 Uložení správné IP adresy

ID = 5A007
Typ testu: Komponentní

Kroky:
1. Jděte na obrazovku nastavení IP adresy
2. Zadejte validní IP adresu
3. Potvrďte zadanou IP adresu
4. Opětovně jděte na obrazovku nastavení IP adresy
5. Zkontrolujte uloženou hodnotu IP adresu

Očekávaný výsledek:
- IP adresa bude úspěšně validováno a uloženo v aplikaci

7.5.8 Uložení IP adresy s prázdnou hodnotou

ID = 5A008
Typ testu: Komponentní
Kroky:
1. Jděte na obrazovku nastavení IP adresy
2. Zadejte prázdnou IP adresu
3. Potvrďte zadanou IP adresu
4. Zkontrolujte chybové hlášení

Očekávaný výsledek:
- IP adresa se neuloží
- Chybové hlášení o nevalidní IP adrese

7.5.9 Uložení krátké a nevalidní IP adresy

ID = 5A009
Typ testu: Komponentní
Kroky:
1. Jděte na obrazovku nastavení IP adresy
2. Zadejte nevalidní IP adresu
3. Potvrďte zadanou IP adresu
4. Zkontrolujte chybové hlášení

Očekávaný výsledek:
- IP adresa se neuloží
- Chybové hlášení o nevalidní IP adrese

7.5.10 Kontrola uložené IP adresy, odchodu bez změn a následného příchodu na obrazovku s IP adresou a provedení její kontroly

ID = 5A0010
Typ testu: Komponentní
Kroky:
1. Jděte na obrazovku nastavení IP adresy
2. Proveďte kontrolu hodnoty IP adresy
3. Odejděte z obrazovky nastavení IP adresy beze změn
4. Opětovně jděte na obrazovku nastavení IP adresy
5. Zkontrolujte na obrazovce zobrazenou IP adresu

Očekávaný výsledek:
- Opětovné zobrazení IP adresy na obrazovce bude obsahovat uloženou hodnotu

7.5.11 Uložení správné hodnoty portu

ID = 5A011
Typ testu: Komponentní
Kroky:
1. Jděte na obrazovku nastavení portu
2. Zadejte validní port
3. Potvrďte zadanou hodnotu portu
4. Zkontrolujte uloženou hodnotu portu

Očekávaný výsledek:
- Port bude úspěšně validovaný a uložený v aplikaci

7.5.12 Uložení portu s prázdnou hodnotou

ID = 5A0012
Typ testu: Komponentní
Kroky:
1. Jděte na obrazovku nastavení portu
2. Zadejte prázdnou hodnotu portu
3. Potvrďte zadanou hodnotu portu
4. Zkontrolujte chybové hlášení

Očekávaný výsledek:
- Prázdná hodnota portu nebude uložena
- Chybové hlášení o nevalidním portu

7.5.13 Uložení vyšší hodnoty portu, než je maximální hodnota portu

ID = 5A0013
Typ testu: Komponentní
Kroky:
1. Jděte na obrazovku nastavení portu
2. Zadejte nevalidní hodnotu portu
3. Potvrďte zadanou hodnotu portu
4. Zkontrolujte uložení hodnoty portu

Očekávaný výsledek:
- Port se uloží
- Zadaná hodnota portu se bude rovnat maximální hodnotě portu

7.5.14 Kontrola uloženého portu, odchodu bez změn a následného příchodu na obrazovku s hodnotou portu a provedení její kontroly

ID = 5A0014
Typ testu: Komponentní

Kroky:

1. Jděte na obrazovku nastavení portu
2. Proveďte kontrolu hodnoty portu
3. Odejděte z obrazovky nastavení portu beze změn
4. Opětovně jděte na obrazovku nastavení portu
5. Zkontrolujte na obrazovce zobrazenou hodnotu portu

Očekávaný výsledek:

- Opětovné zobrazení hodnoty portu na obrazovce bude obsahovat uloženou hodnotu

8 IMPLEMENTACE VYBRANÝCH TESTOVACÍCH SAD

Pro ověření funkčnosti vytvořené metodiky bylo potřeba provést implementaci testovacích sad. Pro implementaci se zvolily sady z kapitoly o obecných testovacích sadách. Před fází testování získané aplikace na platebním terminálu Newland N910, bylo potřeba si testovací sady přizpůsobit.

Testovací sadu pro autentizaci byla rozdělena do dvou testovacích sad. Důvodem byl rozdílný přístup k autentizaci z pohledu zákazníka a servisního technika. Získaná testovací aplikace přistupovala rozdílně k přihlášení a odhlášení zmíněných uživatelů. Rozdílné byly i počáteční podmínky testovacích případů. Zbylé testovací sady byly podle předpisu přímo implementovány.

Při implementaci se vycházelo z výsledků, ke kterým se došlo v kapitole zabývající se technologiemi pro automatizované testování platebních terminálů.

8.1 Architektura vytvořeného projektu pro implementaci testovacích sad

Základem projektu se stal objektově orientovaný programovací jazyk Java 8. Vybraný programovací jazyk byl zvolen na základě jeho rozšířenosti ve firmě, ve které byla vytvořená metodika použita. Pro správu projektu byl využit nástroj Maven a implementace probíhala ve vývojovém prostředí IntelliJ IDEA.

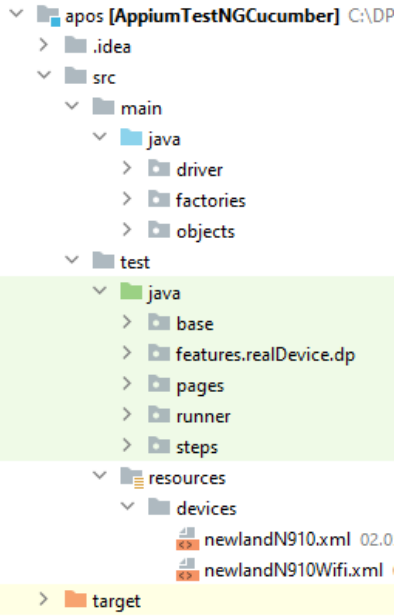
Pro komunikaci s aplikací na platebním terminálu a provádění úkonů z pohledu uživatele bylo zvoleno Appium. Samotné Appium by bylo nedostatečné. Pro snadné porozumění jednotlivých testů se využil Cucumber. Pro zpřehlednění a rozšíření možností běhu projektu se vybrané technologie zaobalily do testovacího frameworku TestNG. Kombinace uvedených technologií lze snadno modifikovat podle aktuálních potřeb.

Při implementaci projektu byla struktura rozdělena na dvě hlavní části. Jedná se o adresáře main a testy, které jsou viditelné na obrázku (Obr. 15).

Obsah adresáře main představuje řídící třídy projektu, které neprovádí testy. Daná část umožňuje vybírat zařízení, na kterém testy poběží. Upravuje ID profilu podle zvoleného uživatele, volí platformu pro běh testů a jaký balíček aplikace bude na zařízení vybraný.

Druhý adresář obsahuje samotné feature soubory, kroky, obrazovky a další nezbytné třídy pro provádění automatizovaných testů. Spadají zde i XML soubory k jednotlivým terminálům, které jsou využívaný pro automatizované testy.

Struktura projektu je navržena tak, aby fungovala jak pro malé, tak i velké projekty. Její použití nemusí být omezeno pouze na testování platebních terminálu, ale lze si ji přizpůsobit i pro testování webových aplikací.



Obrázek 15 Struktura projektu implementovaných testovacích sad

8.2 Implementování testovacího případu

Každá testovací sada je reprezentovaná vlastním feature souborem. Feature soubor se skládá ze scénářů, které představují jednotlivé testovací případy. Na obrázku (Obr. 16) je ukázka implementovaného testovacího případu s ID 5A005. Pomocí jazyku Gherkin jsou jednotlivé kroky ve scénáři čitelné a lehce pochopitelné.

```
Scenario: 5A005 Set invalid profile ID with 7 chars and check error message
  Given Open APOS application
  When click on side menu button on idle screen
  And click on service menu button on side menu screen
  And set value on keyboard on service menu password screen: 0000
  And click on confirm button on service menu password screen
  And click on init menu button on service menu screen
  And click on set init button on init menu screen
  And click on set profile id button on set init screen
  And set profile id on keyboard on set profile id screen: XXXXYYY
  And click on confirm button on set profile id screen
  Then check profile id error message on profile id screen
```

Obrázek 16 Ukázka implementovaného testovacího případu

Obrázek (Obr. 17) představuje příklad, který obsahuje definice jednotlivých kroků mapující každý krok do kódu. Zde dochází k vykonání akce, která je popsaná krokem.

```
@When("^click on refund button on idle screen$")
public void clickOnRefundButton() throws InterruptedException {
    screenObject.clickOnElement(idleScreen.getRefundButton());
}

@When("^set value on idle screen: ([^\"]*)")
public void setValue(String number) throws InterruptedException {
    Thread.sleep( millis: 100);
    screenObject.selectNumbers(splitNumbers(number));
}

@When("click on confirm button on idle screen$")
public void clickOnConfirmButton() throws InterruptedException {
    screenObject.clickOnElement(idleScreen.getConfirmButton());
}
```

Obrázek 17 Příklad definice jednotlivých kroků

Akce nemůže být nad aplikací v terminálu provedena, pokud nedojde k přesné lokalizaci elementů. Tomu dopomáhají třídy, které v projektu představují jednotlivé obrazovky aplikace se všemi elementy. Každý element je zde lokalizovaný pomocí ID nebo Xpath. Pomocí lokalizace daného elementu lze nad ním vyvolat konkrétní akci v definici jednotlivých kroků. Obrázek (Obr. 18) obsahuje ukázku elementů z hlavní obrazovky, které jsou lokalizované pomocí ID.

```
14
15      @AndroidFindBy(id = "com.payten.apos:id/icon_image_view")
16      private AndroidElement imageTitle;
17
18      @AndroidFindBy(id = "com.payten.apos:id/title_text_view")
19      private AndroidElement titleText;
20
21      @AndroidFindBy(id = "com.payten.apos:id/input_edit_text")
22      private AndroidElement editText;
23
24      @AndroidFindBy(id = "com.payten.apos:id/tv_currency_symbol")
25      private AndroidElement currency;
26
```

Obrázek 18 Ukázka lokalizovaných elementů pomocí ID

Výsledek běhu testovací sady reprezentuje zpráva o běhu, která je vygenerovaná pomocí Cucumberu.

8.3 Budoucí rozšíření testovacího projektu

Zmíněnou implementaci testovacího projektu lze aktuálně použít pro běh z lokálního zařízení. Taková forma běhu je značně omezená. Možnost rozšíření běhu například umožňuje Jenkins server. Pro budoucí použití testovacího projektu v reálném provozu je spojení projektu a Jenkinse stěžejní.

Předpokládá se, že projekt již běží na verzovacím systému. Pro Jenkins je potřeba zřídit přístup do projektu uloženého na verzovacím systému. Následně lze na Jenkinsi nastavit projekt podle našich potřeb.

V Jenkinsi lze následně sledovat výsledky běhu testů v průběhu času. To umožňuje sledovat celkovou kondici vyvíjené aplikace a lze usoudit, zda vývoj probíhá podle určeného plánu.

9 ZHODNOCENÍ DOSAŽENÝCH VÝSLEDKŮ

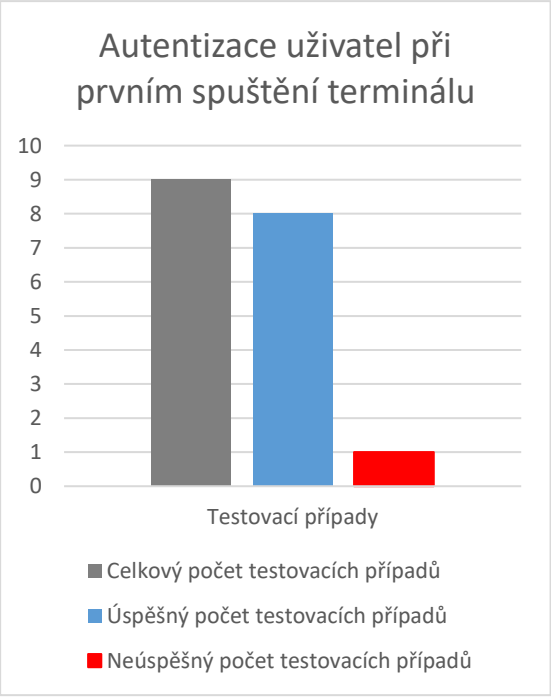
Pro prezentování vytvořené metodiky bylo implementováno pět testovacích sad, které jsou detailněji rozvedeny v kapitole 7.

Jedná se o sady:

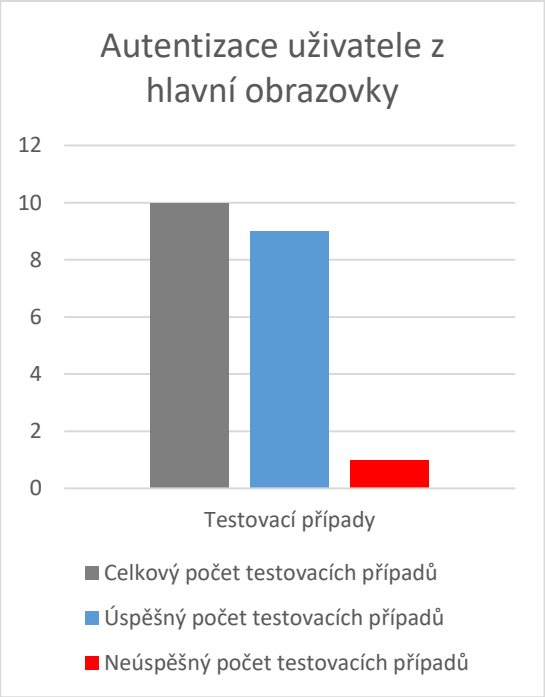
1. Autentizace
2. Validace vstupních polí aplikace
3. Kontrola obrazovek
4. Inicializace terminálu
5. Nastavení IP adresy, portu, ID profilu, SSL

9.1 Autentizace

Testovací sada pro kontrolu autentizace uživatele do servisního menu je rozdělena na další dvě sady. Jedná se o autentizaci uživatel při prvním spuštění terminálu a autentizace uživatel z hlavní obrazovky aplikace. Obě sady testují stejnou přihlašovací obrazovku, ale za jiných počátečních podmínek. Pro odhlášení uživatele je použit časový limit pouze jen u autentizace uživatele z hlavní obrazovky aplikace.



Graf 2 *Výsledek sady autentizace uživatele při prvním spuštění*



Graf 1 *Výsledek sady autentizace uživatele z hlavní obrazovky*

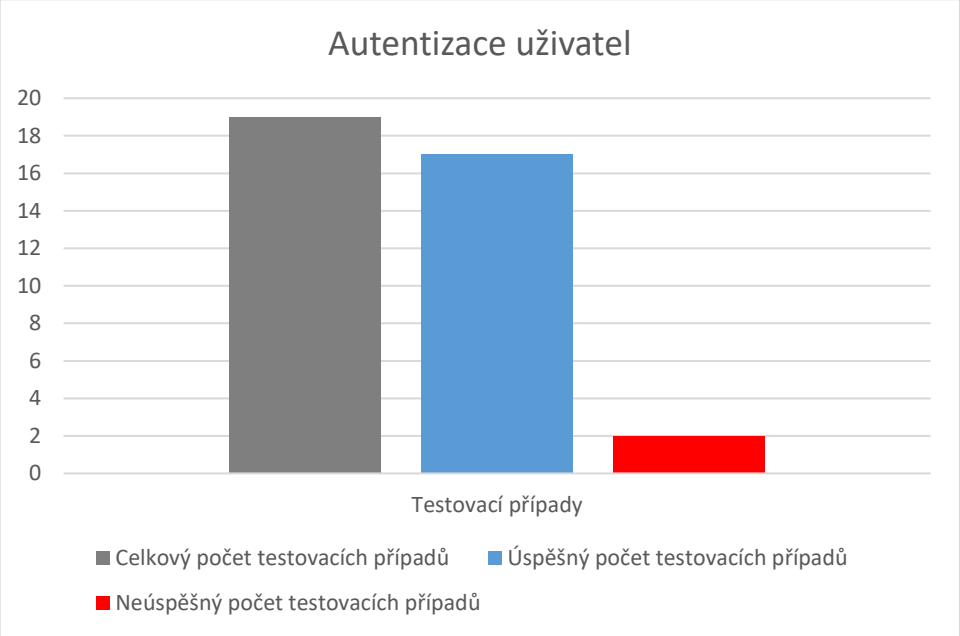
Výsledek běhu testovací sady pro autentizaci uživatele z hlavní obrazovky je vyobrazen na grafu (Graf 2). Výpočet úspěšnosti testovací sady dostaneme počtem úspěšných testů děleno celkovým počtem testů (Rov. 1), která dosáhla 90 %. Neúspěšnost testů je zde na hodnotě 10 %.

$$\frac{9}{10} * 100 = 90\ \% \tag{1}$$

Výsledek běhu testovací sady pro autentizaci uživatele při prvním spuštění terminálu je vyobrazen na grafu (Graf 1). Výsledek úspěšnosti testovací sady je vypočítaný pomocí rovnice (Rov. 2). Neúspěšnost dosahovala 11,11 %.

$$\frac{8}{9} * 100 = 88{,}89\ \% \tag{2}$$

Celkový výsledek z běhu testovacích případů pro autentizaci uživatele je vyobrazen na grafu (Graf 3). Celková úspěšnost testovacích případů je zde na 89,47 %. Neúspěšnost testovacích případů je 10,53 %.

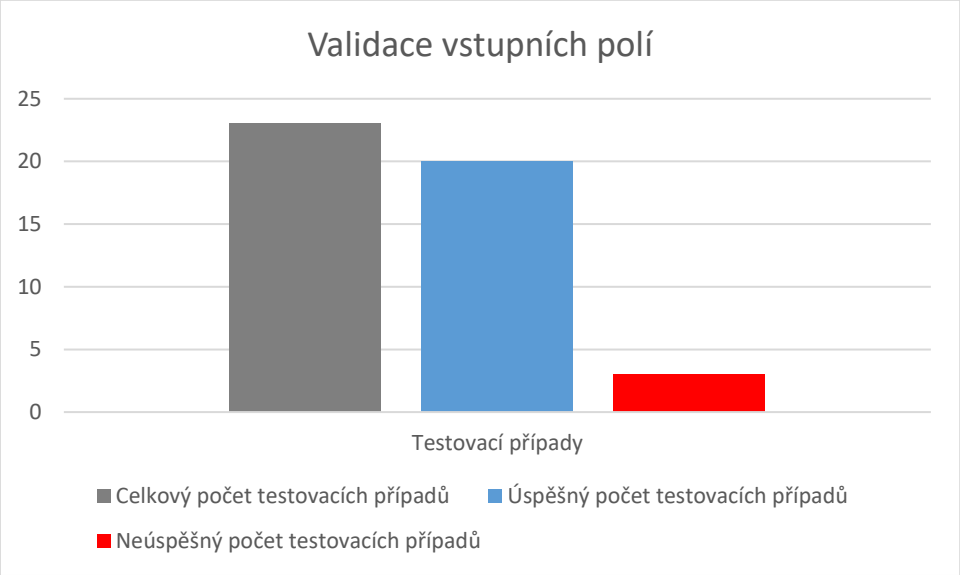


Graf 3 Výsledky testovacích sad autentizace uživatel

Po prozkoumání chyb z testovacích sad se došlo k závěru, že chyba není způsobena špatnou implementací testu, ale chybou v aplikaci. Jedná se o opakované neúspěšné přihlášení do platebního terminálu. Po třech pokusech by mělo dojít k blokaci a zobrazení chybové hlášky, která o dané skutečnosti informuje. Aplikace umožňuje neomezený počet pokusů pro zadání hesla do servisní části terminálu. Zmíněné chyby představují bezpečností riziko přístupu do servisního menu.

9.2 Validace vstupních polí aplikace

Testovací sada validace vstupních polí aplikace obsahuje celkem 23 testovacích případů. Během těchto případů jsou testovány všechny vstupní pole, které se v poskytnuté aplikaci nachází. Vstupem byly různé kombinace hodnot, kterými se validace ověřovala.



Graf 4 Výsledek testovací sady validace vstupních polí

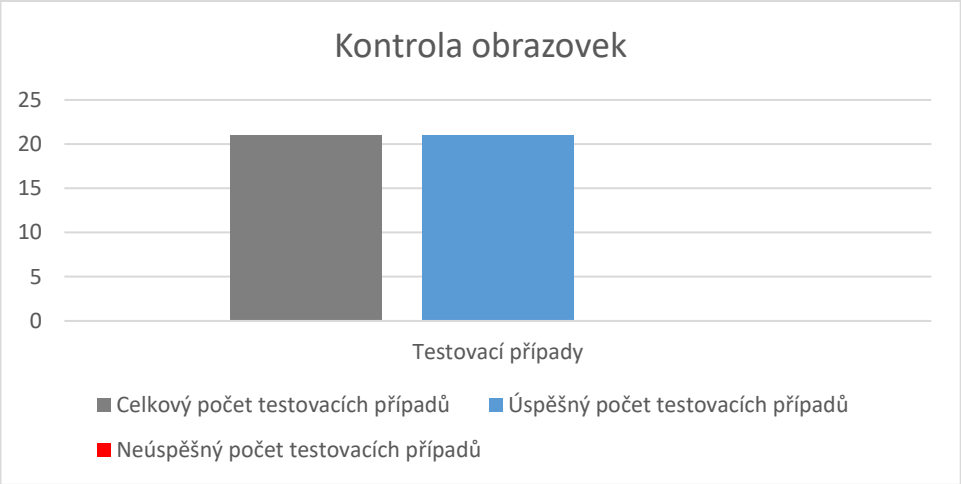
Z 23 testovacích případů jich bylo 20 úspěšných. Úspěšnost běhu testovací sady lze ohodnotit na 86,96 %. Neúspěšnost představují 3 testovací případy. Procentuálně se jedná o 13,04 % neúspěšných testovacích případů.

Chyby, které byly zachyceny testovacími případy jsou spojeny se zadáváním hodnoty portu. Objevila se nefunkčnost omezení velikosti čísla portu a zadání nečíselné hodnoty. Zmíněné chyby lze téměř považovat za estetické. Neovlivňují funkci aplikace a nacházejí se v servisním menu, kam běžní uživatelé nemají přístup.

9.3 Kontrola obrazovek

Testovací sada pro kontrolu obrazovek aplikace terminálu obsahuje celkem 21 testovacích případů. V testovací sadě docházelo ke kontrole elementů, které měly být zobrazeny na jednotlivých obrazovkách.

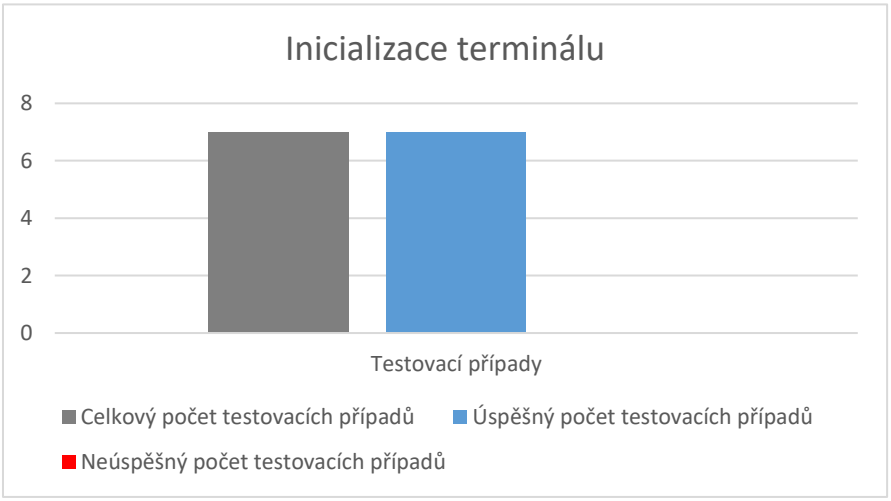
Výsledek testovací sady je zřejmý z grafu (Graf 5). Testovací sada proběhla bez zaznamenání chyby. Výsledek úspěšnosti představuje 100 % a neúspěšnosti 0 %.



Graf 5 Výsledek testovací sady kontrola obrazovek

9.4 Inicializace terminálu

Inicializace aplikace platebního terminálu představuje čtvrtou implementovanou testovací sadu platebního. Sada obsahuje pozitivní i negativní testovací případy.



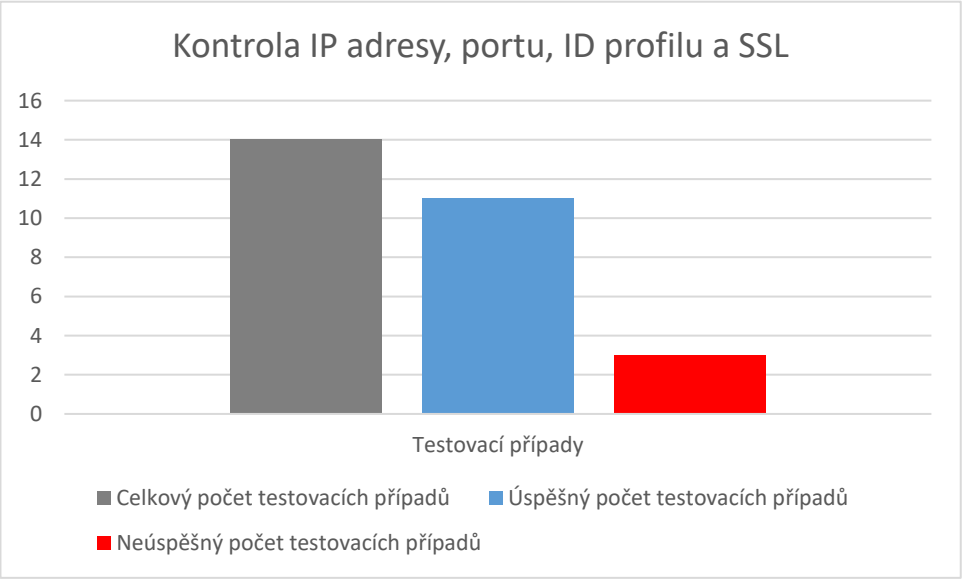
Graf 6 *Výsledek testovací sady inicializace terminálu*

Výsledek běhu testovací sady je znázorněn na grafu (Graf 6). Úspěšnost běhu sady představuje 100 % testovacích případů. Z pohledu implementovaných testů pro inicializaci aplikace platebního terminálu nebyla nalezena chyba.

9.5 Nastavení IP adresy, portu, ID profilu, SSL

Poslední testovací sadu představuje nastavení IP adresy, portu, ID profilu a SSL. Testovací sada neřeší validaci jednotlivých polí, ale řeší nastavení a ukládání těchto hodnot.

Výsledek testovací sady je patrný z grafu (Graf 7). Celkový počet úspěšných testovacích případů je 11 ze 14 celkových. Procentuálně se jedná o 78,57 % úspěšných testovacích případů. Zbylých 21,43 % jsou neúspěšné testovací případy.



Graf 7 Výsledek testovací sady kontroly IP adresy, portu, ID profilu a SSL

Po analýze objevených chyb byla zjištěna jedna chyba u portu a dvě chyby u ukládání ID profilu.

U portu lze uložit číslo o velikosti 99999. Nejedná se o kritickou chybu, protože následná inicializace na zadané číslo není schopna komunikovat a je ukončena chybovou hláškou o špatně zadaných parametrech.

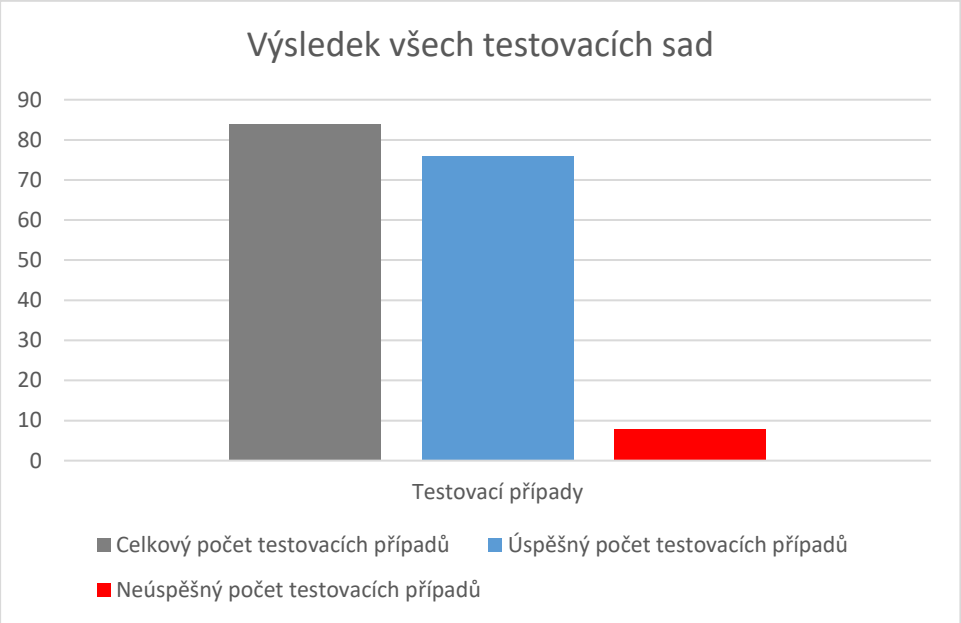
Zbylé dvě chyby jsou spojeny s ID profilu. Ukládání zadaných hodnot nefunguje vždy správně. Pokud chce uživatel odejít beze změny uloženého ID, objeví se chybové hlášení,

které signalizuje špatně zadané ID profilu. Druhá situace je při vkládání prázdné hodnoty ID profilu. Obě chyby lze považovat za méně závažné.

9.6 Celkové zhodnocení dosažených výsledků

Po dílčím zhodnocení testovacích sad je potřeba výsledky spojit dohromady. Výsledky testovacích sad nelze porovnávat v průběhu času na různých verzí aplikace pro platební terminály. K dostání je pouze jediná verze APOS. Je možné zhodnotit pouze aktuálně získané výsledky.

Z grafu (Graf 8) je zřejmé, že výsledek se skládá z 84 testovacích případů, které jsou rozprostřeny do jednotlivých testovacích sad. Osm testovacích případů představují objevené chyby a 76 testovacích případů proběhlo úspěšně.

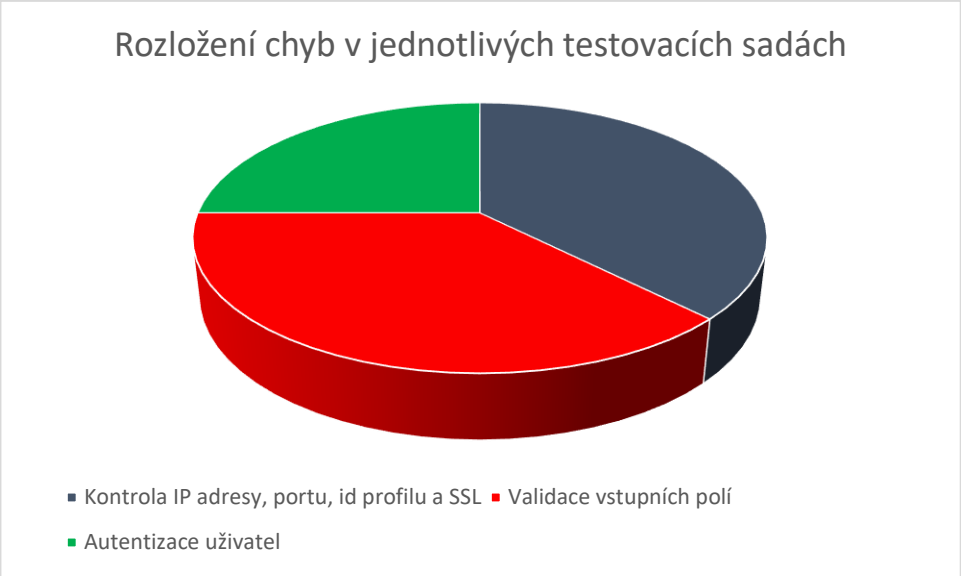


Graf 8 Výsledek běhu implementovaných testovacích sad

Úspěšnost testovacích případů je vypočtena na rovnici (Rov. 3). Neúspěšnost je dopočítaná na 9,52 %.

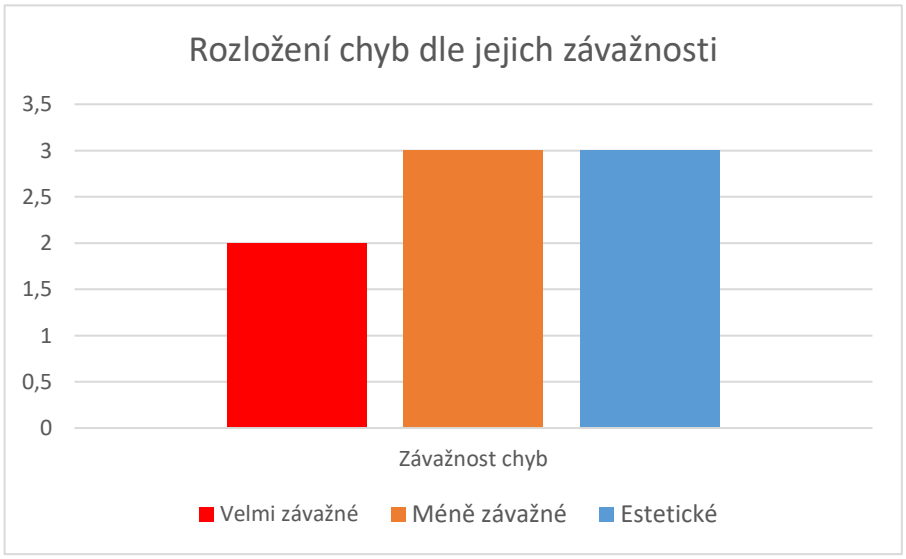
$$\frac{76}{84} * 100 = 90{,}48\ \% \tag{3}$$

Graf (Graf 9) zobrazuje testovací sady, ve kterých došlo k nalezení chyb. Při větším počtu testovacích sad a nalezených chyb, by lépe vyplynulo rozložení chyb a oblasti aplikace, které budou vyžadovat nejvíce práce na uvedení do požadovaného stavu.



Graf 9 Rozložení chyb v jednotlivých testovacích sadách

Graf (Graf 10) zobrazuje počet nalezených chyb, které jsou rozděleny do skupin podle jejich závažnosti. Podle vyobrazených informací se objevily dvě velmi závažné chyby, které znemožňují danou verzi aplikace předat zákazníkovi. Jedná se o opakované neúspěšné přihlášení uživatele do servisního menu a následnou blokaci nebo časové prodlení pro opětovné zadávání hesla. Po opravě velmi závažných chyb lze aplikaci zákazníkovi předat. Chyby s nižší prioritou lze opravit později, protože nepředstavují bezpečností nebo funkční rizika aplikace.



Graf 10 Rozložení chyb dle jejich závažnosti

ZÁVĚR

Hlavním cílem diplomové práce bylo navrhnout vlastní metodiku pro tvorbu automatizovaných testů platebních terminálů na platformě Android. Aby bylo možné navrhnout vhodnou metodiku, bylo nutné se nejdříve podrobně seznámit s testováním aplikací na platformě Android, technologiemi pro automatizované testování a principem aplikací pro platební terminály. Samozřejmostí bylo získání vhodného platebního terminálu s platební aplikací, na které se bude moct provést test navržené metodiky. Nedílnou součásti získávání znalostí bylo manuální testování aplikace na propůjčeném terminálu.

Díky získaným poznatkům bylo možné sestavit vlastní metodiku a ověřit ji na aplikaci pro platební terminály. Bylo připraveno pět testovacích sad s testovacími případy, které byly obsaženy v navržené metodice. Na základě vybraných technologií se sestavil projekt a implementovaly se testovací sady. Na závěr byly představeny výsledky, ke kterým se při testování dospělo.

Metodika byla navržena pro pokrytí všech základních kritických oblastí aplikací pro platební terminály. Získané výsledky dokazují, že testovací případy navržené podle metodiky jsou funkční. Odhalení chyb různých závažností ve zkušební aplikaci jsou toho důkazem.

Navržená metodika je již uplatněna při automatizovaném testování platebních aplikací ve společnosti, která platební terminál s testovací verzí aplikace zapůjčila. Nedílnou součástí je i následné rozšíření metodiky na základě požadovaných funkcionalit aplikace, které jsou úzce specifické pro podnikatelský záměr zákazníka.

SEZNAM POUŽITÉ LITERATURY

[1]JAMIL, Muhammad Abid and col. *Software Testing Techniques*: A Literature Review. 6th International Conference on Information and Communication Technology for The Muslim World, 2016. Dostupné z: DOI: 10.1109/ICT4M.2016.045

[2]LI, Kanglin and WU, Mengqi. *Effective Software Test Automation—Developing an Automated Software Testing Tool*. CA: Sybex, 2004. ISBN:0782143202

[3]Geeksforgeeks: [online]. [cit. 2021-01-15]. Dostupné z: https://www.geeksforgeeks.org

[4][HEDAOO, Avinash H. and KHANDELWAL, Abha. *Study of Dynamic Testing Techniques.* International Journal of Advanced Research in Computer Science and Software Engineering. [online]. April 2017, vol. 7, iss. 4, s. 322-330 [cit. 2020-11-16]. ISSN: 2277 128X. Dostupné z: 10.23956/ijarcsse/V7I4/0136

[5]CRAIG, Rick and JASKIEL, Steve. *Systematic Software Testing.* Artech House, 2002. ISBN: 1580535089.

[6]JORGENSEN, Paul C. *Software Testing:* A Craftsman's Approach. 4th ed. New York: Taylor & Francis Group, 2014. ISBN: 978-1-4665-6068-0.

[7]PATTON, Ron. *Testování softwaru*. 1. vyd. Praha: Computer Press, 2002. ISBN: 80-7226-636-5.

[8]SORENSEN, Emily: *The historical roots of electronic card machines*. In: Mobiletransaction [online]. 26. 7. 2019 [cit. 2021-02-02]. Dostupné z: https://www.mobiletransaction.org/history-of-credit-card-machines/

[9]Moneris: [online]. ©2021 [cit. 2020-01-15]. Dostupné z: https://insights.moneris.com/business-matters/how-did-we-get-here-the-history-of-payment-terminals

[10]PEARCE, Bryony. *The history of card machines.* In: Takepayments [online]. 11. 10. 2019 [cit. 2021-02-08]. Dostupné z: https://www.takepayments.com/blog/product-information/the-history-of-card-machines/

[11]ESTEP, Jamie. *The History of Credit Card Terminals.* In: Merchantequip [online]. 12. 4. 2006 [cit. 2021-02-08]. Dostupné z: https://www.merchantequip.com/merchant-account-blog/102/the-history-of-credit-card-terminals

[12] DIVIN, Nicolas: Why Android is the future of POS. In: Ingenico [online]. 10. 10. 2018 [cit. 2021-02-12]. Dostupné z: https://blog.ingenico.com/posts/2018/09/why-android-is-the-future-of-pos.html

[13] EMVCo. EMV Contactless Specifications for Payment Systems. In Book A: Architecture and General Requirements v2.6. EMVCo, 2016. Dostupné z: https://www.emvco.com/wp-content/uploads/2017/05/Book_A_Architecture_and_General_Rqmts_v2_6_Final_20160422011856105.pdf

[14] Abrantix: [online]. Abrantix AG, © 2021 [cit. 2021-01-15]. Dostupné z: https://www.abrantix.com/en/payment-test-automation.html

[15] KRAJCI, Iggy and CUMMINGS, Darren. *Android on x86: An Introduction to Optimizing for Intel Architecture*. Berkeley: Apress, 2013. ISBN: 978-1-4302-6130-8.

[16] Developer.android: [online]. ©2021 [cit. 2020-02-13]. Dostupné z: https://developer.android.com/guide/platform

[17] KNOTT, Daniel. *Hands-on mobile app testing: a guide for mobile testers and anyone involved in the mobile app business*. Crawfordsville: Addison-Wesley Professional, 2015. ISBN: 9780134191829

[18] BRADSHAW, Kyle and ROUSSEL, Dylan. Google kills Android distribution numbers on the web, but we've got you covered. In: *9to5google.* [online]. Apr 10, 2020. [cit. 2021-01-13]. Dostupné z: https://9to5google.com/2020/04/10/google-kills-android-distribution-numbers-web/

[19] Appium: [online]. JS Foundations, ©2021 [cit. 2021-01-18]. Dostupné z: http://appium.io/

[20] GARG, Shankar. *Appium Recipes*. CA: Apress, 2016. ISBN: 978-1-4842-2417-5

[21] Digital: [online]. Digital.ai. Software Inc., © 2021 [cit. 2021-01-18]. Dostupné z: https://experitest.com/selenium-testing/web-and-mobile-test-automation-drivers/

[22] TestProject: [online]. © 2021 [cit. 2020-12-15]. Dostupné z: https://testproject.io/

[23] GitHub: [online]. GitHub, Inc. ©2021 [cit. 2020-11-06]. Dostupné z: https://github.com/RobotiumTech/robotium

[24] Apache: [online]. The Apache Software Foundation, ©2019 [cit. 2021-02-09]. Dostupné z: https://www.apache.org/licenses/LICENSE-2.0 rok 2004

[25] HELLESØY, Aslak. The world's most misunderstood collaboration tool. In:Cucumber [online]. 3. 4. 2014 [cit. 2020-12-19]. Dostupné z: https://cucumber.io/blog/collaboration/the-worlds-most-misunderstood-collaboration-tool/

[26] Cucumber: [online]. SmartBear Software, © 2019 [cit. 2020-12-03]. Dostupné z: https://cucumber.io/docs/guides/overview/

[27] SEB, Rose; WYNNE, Matt; HELLESOY, Aslak. *The Cucumber for Java Book: Behaviour-Driven Development for Testers and Developers*. Dallas: Pragmatic Bookself, 2015. ISBN-13: 978-1941222294

[28] TestNG: [online]. Cédric Beust, ©2019 [cit. 2020-11-03]. Dostupné z: https://testng.org/doc/documentation-main.html

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

API	*Application Programming Interface*
APOS	*Android Point of Sale*
ATDD	*Acceptance test driven development*
BDD	*Behavior-driven development*
C#	*C Sharp*
CPU	*Central processing unit* (centrální procesorová jednotka)
CSS	*Cascading Style Sheets* (kaskádové styly)
GPS	*Global Positioning System* (globální polohový systém)
GPU	*Graphics processing unit (grafický procesor)*
GUI	*Graphical User Interface* (grafické uživatelské rozhraní)
HTML	*Hypertext Markup Language*
IMB	*International Business Machines Corporation*
JSON	*JavaScript Object Notation*
LTE	*Long Term Evolution*
NFC	*Near field communication*
PAN	*Primary account number*
PHP	*Hypertext Preprocessor*
PIN	*Personal identification number*
PIN pad	*PIN entry device*
POS	*Point of sale* (prodejní místo)
QA	*Quality assurance*
SDK	*Software Development Kit*
SSL	*Secure Sockets Layer*
UDID	*Unique Disability ID* (jedinečný identifikátor zařízení)

UI	*User interface* (uživatelské rozhraní)
URL	*Uniform Resource Locator* (jednotný lokátor zdroje)
UWP	*Universal Windows Platform*
XML	*Extensible Markup Language*

SEZNAM OBRÁZKŮ

SEZNAM TABULEK A GRAFŮ

SEZNAM PŘÍLOH

PŘÍLOHA P I: OBSAH CD

Struktura obsahu přiloženého CD:

- **DP_Lences-Marian.pdf** – obsahuje text diplomové práce ve formátu PDF.
- **prilohy.zip** – obsahuje přiložený projekt v adresáři **Projekt** a výsledky běhu implementovaných testovacích sad v adresáři **Reporty**

PŘÍLOHA P II: STROM VAZEB UKÁZKOVÝCH TESTOVACÍCH SAD

1 - Testování aplikace pro platební terminály

1A - Autentizace

1A001 - Kontrola validní autentizace do servisního menu

1A002 - Kontrola autentizace bez zadání hesla do servisního menu

1A003 - Kontrola autentizace se zadáním hesla o minimální délce počtu znaků do servisního menu

1A004 - Kontrola autentizace se zadáním hesla o maximální délce počtu znaků do servisního menu

1A005 - Kontrola autentizace se zadáním hesla o jedno delší, než je maximum do servisního menu

1A006 - Kontrola autentizace se zadáním nevalidních znaků pro přihlášení do servisního menu

1A007 - Kontrola maskování zadaných čísel hesla do servisního menu

1A008 - Odhlášení uživatele ze servisního menu

1A009 - Opakované neúspěšné zadávání hesla do servisního menu

1A010 - Odhlášení uživatele ze servisního menu v případě nečinnosti 30 sekund

2A - Validace vstupních polí

2A001 - Zadání validní celočíselné částky pro transakci prodej

2A002 - Zadání validní částky s desetinou čárkou pro transakci prodej

2A003 - Zadání nulové částky pro transakci prodej

2A004 - Validace celočíselné částky o jedna větší, než je maximum pro transakci prodej

2A005 - Validace částky s několika desetinnými čárky, celočíselnou hodnotou vyšší, než je maximum a 3 desetinnými pozicemi pro transakci prodej

2A006 - Validace vložení nečíselného řetězce pro transakci prodej

2A007 - Zadání validní celočíselné částky pro transakci návrat

2A008 - Zadání validní částky s desetinou čárkou pro transakci návrat

2A009 - Zadání nulové částky pro transakci návrat

2A010 - Validace celočíselné částky o jedna větší, než je maximum pro transakci návrat

2A011 - Validace částky s několika desetinnými čárky, celočíselnou hodnotou vyšší, než je maximum a 3 desetinnými pozicemi pro transakci návrat

2A012 - Validace vložení nečíselného řetězce pro transakci návrat

2A013 - Validace existujícího portu

2A014 - Validace nejnižšího čísla portu

2A015 - Validace nevyššího čísla portu

2A016 - Validace o jedno vyšší číslo, než je maximum čísla portu

2A017 - Validace čísla portu s hodnotou 999999

2A018 - Validace nečíselného řetězce portu

2A019 - Validace validního ID profilu

2A020 - Validace ID profilu s větší délkou ID, než je povoleno

2A021 - Validace správné IP adresy

2A022 - Validace nevalidní IP adresy

2A023 - Validace textového řetězce místo IP adresy

3A - Kontrola obrazovek

3A001 - Kontrola elementů na hlavní obrazovce

3A002 - Kontrola elementů na bočním menu

3A003 - Kontrola elementů na transakčním menu

3A004 - Kontrola elementů pro transakci prodej

3A005 - Kontrola elementů pro transakci návrat

3A006 - Kontrola elementů pro menu dávky

3A007 - Kontrola elementů pro menu nastavení

3A008 - Kontrola elementů nastavení jazyku

3A009 - Kontrola elementů na přihlašovací obrazovce do servisního menu

3A010 - Kontrola elementů servisního menu

3A011 - Kontrola elementů správy klíčů

3A012 - Kontrola elementů inicializačního menu

3A013 - Kontrola elementů pro menu nastavení inicializace

3A014 - Kontrola elementů nastavení portu

3A015 - Kontrola elementů nastavení IP adresy

3A016 - Kontrola elementů ID profilu

3A017 - Kontrola elementů nastavení SSL

3A018 - Kontrola elementů informace terminálu

4A - Inicializace terminálu

4A001 - Úspěšné provedení inicializace ze servisního menu

4A002 - Úspěšné provedení inicializace z uživatelského nastavení

4A003 - Inicializace s neexistujícím ID profilu

4A004 - Inicializace se špatnou IP adresou

4A005 - Inicializace se špatným portem

4A006 - Inicializace se zakázaným SSL

4A007 - Inicializace bez připojení k internetu

5A - IP adresa, port, ID profilu, SSL

5A001 - Kontrola povolení SSL

5A002 - Kontrola zakázaní SSL

5A003 - Uložení validního ID profilu

5A004 - Uložení ID profilu s prázdným ID

5A005 - Kontrola uložení nesprávné délky ID profilu

5A006 - Kontrola uloženého ID profilu, odchodu bez změn a následného příchodu na obrazovku s ID profilem a provedení jeho kontroly

5A007 - Uložení správné IP adresy

5A008 - Uložení IP adresy s prázdnou hodnotou

5A009 - Uložení krátké a nevalidní IP adresy

5A010 - Kontrola uložené IP adresy, odchodu bez změn a následného příchodu na obrazovku s IP adresou a provedení její kontroly

5A011 - Uložení správné hodnoty portu

5A012 - Uložení portu s prázdnou hodnotou

5A013 - Uložení vyšší hodnoty portu, než je maximální hodnota portu

5A014 - Kontrola uloženého portu, odchodu bez změn a následného příchodu na obrazovku s hodnotou portu a provedení její kontroly