Univerzita Pardubice

Dopravní fakulta Jana Pernera

Akvizitor ve společnosti LOG-IN CZ s.r.o.

Nikita Sashchenko

Bakalářská práce

2025

Univerzita Pardubice
Dopravní fakulta Jana Pernera
Akademický rok: 2024/2025

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Nikita Sashchenko**
Osobní číslo:	**D21061**
Studijní program:	**B1041A040002 Technologie a management v dopravě**
Specializace:	**Dopravní management a marketing**
Téma práce:	**Akvizitor ve společnosti LOG-IN CZ s.r.o.**
Zadávající katedra:	**Katedra dopravního managementu, marketingu a logistiky**

Zásady pro vypracování

Bakalářská práce zkoumá možnost nahrazení akvizitora ve společnosti LOG-IN CZ s.r.o. umělou inteligencí v kombinaci s online nástroji marketingu. Práce zanalyzuje efektivitu AI nástrojů při získávání nových zákazníků a porovná je s tradičními metodami akvizice. Na základě získaných dat bude posouzeno, zda je možné dosáhnout stejných nebo lepších výsledků pomocí AI.

Rozsah pracovní zprávy: **35-45 stran**
Rozsah grafických prací: **dle doporučení vedoucí/ho**
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

dle pokynů vedoucí/ho práce

Vedoucí bakalářské práce: **Ing. Stefan Jovčić, Ph.D.**
Katedra dopravního managementu, marketingu a logistiky

Datum zadání bakalářské práce: **31. října 2024**
Termín odevzdání bakalářské práce: **27. června 2025**

L.S.

doc. Ing. Ladislav Řoutil, Ph.D.
děkan

Ing. Pavla Lejsková, Ph.D.
vedoucí katedry

V Pardubicích dne 18. června 2025

Prohlašuji:

Práci s názvem akvizitor ve společnosti LOG-IN CZ s.r.o. jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 7/2019 Pravidla pro odevzdávání, zveřejňování a formální úpravu závěrečných prací, ve znění pozdějších dodatků, bude práce zveřejněna prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 27. 6. 2025

Nikita Sashchenko v. r.

Touto cestou bych rád poděkoval vedoucímu bakalářské práce **Ing. Stefan Jovčić, Ph.D.** za odborné vedení, cenné připomínky a podporu během zpracování práce. Dále děkuji společnosti **LOG-IN CZ s.r.o.** za poskytnutí potřebných dat a za vstřícnost při konzultacích.

ANOTACE

Tato bakalářská práce zkoumá možnost nahrazení akvizitora ve společnosti LOG-IN CZ s.r.o. umělou inteligencí v kombinaci s online nástroji marketingu. Práce analyzuje efektivitu AI nástrojů při získávání nových zákazníků a porovnává je s tradičními metodami akvizice. Na základě získaných dat bude posouzeno, zda je možné dosáhnout stejných nebo lepších výsledků pomocí AI. Cílem je poskytnout doporučení pro implementaci AI řešení v oblasti akvizice zákazníků.

KLÍČOVÁ SLOVA

Akviziční nástroje, umělá inteligence, logistické služby, automatizace marketingu, generování leadů, firemní zákazníci

TITLE

Acquisitor in the company LOG-IN CZ s.r.o.

ANNOTATION

This bachelor thesis examines the possibility of replacing the acquirer in LOG-IN CZ s.r.o. with artificial intelligence in combination with online marketing tools. The thesis analyses the effectiveness of AI tools in acquiring new customers and compares them with traditional acquisition methods. Based on the data obtained, it will be assessed whether it is possible to achieve the same or better results using AI. The goal is to provide recommendations for implementing AI solutions in customer acquisition.

KEYWORDS

Acquisition tools, artificial intelligence, logistics services, marketing automation, lead generation, business customers

OBSAH

ÚVOD

Logistický sektor v České republice i střední Evropě prochází zrychlenou digitalizací. Se sílící konkurencí, rostoucími náklady na přepravu a tlakem na efektivitu se schopnost rychle získávat nové klienty stává klíčovým faktorem růstu. Firmy proto hledají způsoby, jak automatizovat rutinní obchodní procesy a škálovat Business to Business (B2B, obchodní vztah mezi firmami) akvizici bez nutnosti výrazného navyšování personálních kapacit.

Zároveň prudce roste využití umělé inteligence (AI) a nástrojů robotické automatizace procesů pomocí robotické procesní automatizace (RPA). Automatizace generování a ověřování leadů se tak posouvá z okrajového „nice to have“ do pozice standardního řešení, které zajišťuje konzistentní výkon obchodních týmů.

Digitální kanály jako e-mail a sociální sítě vykazují v B2B prodeji vysokou účinnost. Například míra otevření obchodních e-mailů dosahuje v Evropská unie (EU) přes 40 %, a to i přes rostoucí vliv nástrojů na ochranu soukromí (např. Apple Mail Privacy). Firmy proto stále častěji propojují e-mailové kampaně s LinkedIn nebo messengery, a vytvářejí automatizované akviziční trychtýře.

Zároveň je nezbytné dbát na právní soulad s regulací General Data Protection Regulation (GDPR, obecné nařízení o ochraně osobních údajů). Nová doporučení Evropského sboru pro ochranu osobních údajů (EDPB) (1/2024) specifikují, jak lze využívat oprávněný zájem při přímém marketingu. V důsledku toho firmy musí automatizovat nejen sběr kontaktů, ale i právní filtry - např. vyloučení osobních adres, validace e-mailů a možnost snadného odhlášení.

Společnost LOG-IN CZ je typickým zástupcem středně velké české logistické firmy. Rychlý růst klientské základny narazil na limity obchodního týmu - firma nemá dedikovaného akvizitora a obchodní zástupci zvládají jen omezený počet interakcí. Proto vznikla potřeba navrhnout a otestovat systém, který:

- využívá Telegram bota s AI příkazy pro vyhledávání klientů (Google Maps, Europages, LinkedIn),
- automatizuje e-mailové kampaně s dynamickou personalizací a sledováním metrik (open rate, míry prokliku (CTR – Click-Through Rate)),
- zajišťuje GDPR compliance „privacy by design“.

Téma úzce souvisí s výzkumnými oblastmi Univerzity Pardubice - řízení logistických řetězců a aplikovaná AI. Práce přináší tři typy přínosu:

- **Vědecký**: ukazuje, jak AI (ChatGPT, Make) mění tradiční B2B trychtýř.
- **Praktický**: navrhuje konkrétní opakovatelný model digitální akvizice.
- **Socioekonomický**: zvyšuje výkonnost Small and Medium-sized Enterprises (SME, malé a střední podniky) firem v oblasti exportní logistiky.

Cílem práce je navrhnout a implementovat systém automatizované akvizice zákazníků s využitím AI nástrojů, e-mailového marketingu a sociálních sítí. Výsledný systém bude testován ve společnosti LOG-IN CZ a hodnocen podle klíčových metrik: počet získaných kontaktů, míra doručení, otevření a odezvy, ale také dle souladu s právními požadavky.

Objektem výzkumu je proces B2B akvizice v logistice, předmětem konkrétní technické řešení (Telegram bot, Make scénáře, subdoména mail.login.cz), které tento proces automatizuje. Metodologie kombinuje analýzu existujících přístupů, návrh řešení, jeho technickou realizaci a měření dopadů.

1 TEORETICKÁ VÝCHODISKA AKVIZICE ZÁKAZNÍKŮ V LOGISTICE

Akvizitor (z lat. *acquisitor* – „získavatel“) je zaměstnanec nebo zprostředkovatel v dopravních či spedičních firmách, jehož hlavním úkolem je získávání nových zákazníků a zakázek. V současné B2B logistice má však širší roli než klasický obchodník:

- vytváření databáze potenciálních klientů (prospecting),
- kvalifikace potřeb (objem, trasy, požadavky na služby),
- příprava nabídek a kalkulace sazeb,
- doprovod obchodních jednání až po uzavření smlouvy,
- předání klienta operativnímu oddělení a následný rozvoj účtu (*account development*).

Akvizitor tak v praxi kombinuje role **business development manažera**, **sales huntera** a částečně **account manažera**. Jeho výkon se měří počtem kvalifikovaných leadů, konverzním poměrem ke smlouvám a **životní hodnotou klienta (CLV)**.

Klasický B2B akviziční trychtýř

Většina odborných studií (např. RAIN Group, 2023) rozlišuje šest základních fází B2B trychtýře – od prvního kontaktu po rozvoj spolupráce:

1. **Prospecting** – vyhledávání a zaujetí cílových firem
2. **Needs Discovery** – zjištění a formulace potřeby
3. **Solution Crafting** – návrh individuálního logistického řešení
4. **Solution Presentation** – prezentace řešení a zdůvodnění přínosů
5. **Win/Contract** – dohoda o podmínkách, podpis smlouvy
6. **Account Development** – rozšíření spolupráce, udržení klienta

V logistice je tento cyklus zpravidla delší: běžně trvá 3–6 měsíců, účastní se více osob, a po podpisu smlouvy následuje technické „onboarding“ – integrace systémů, zkušební přepravy atd.

Tabulka 1 Fáze akvizičního trychtýře v logistice

Fáze (logistický důraz)	Klíčové činnosti akvizitora	Výstup
0. Market Mapping	Statistická analýza trhu, sestavení „dlouhého seznamu“ z databází (Europages, Firmy.cz, CargoClub aj.)	Segmentovaný seznam potenciálních zákazníků
1. Prospecting	Scraping kontaktů, cold e-mail/LinkedIn/Facebook outreach, Telegram bot /maps a /search	Prvotní zájem (otevření e-mailu, kliknutí, „napište víc“)
2. Qualification	Hovor/chat, upřesnění objemů, tras, cenové citlivosti	Ověřený kvalifikovaný lead (SQL)
3. Solution Design	Kalkulace cen, návrh tarifů, SLA, tvorba unikátní obchodní nabídka (UTP) (/utp)	Personalizovaná obchodní nabídka
4. Negotiation	Jednání o podmínkách, právní kontrola, dohoda na klíčových ukazatelů výkonnosti (KPI)	Ústní nebo písemná dohoda
5. Contract & Onboarding	Podpis, výměna dat přes EDI (Electronic Data Interchange) a API (Aplikační programové rozhraní), zkušební přepravy	Aktivní klient
6. Operations & Review	Sledování prvních zásilek, sběr zpětné vazby	Potvrzená spokojenost
7. Account Growth	Cross-/upsell, žádosti o doporučení, účast ve výběrových řízeních	Růst obratu, CLV

Zdroj: upraveno podle Rackham (1988), Christopher (2016), vlastní úprava autora.

Využití digitálních kanálů, jako jsou e-mail a sociální sítě, výrazně urychluje počáteční fáze obchodního procesu (fáze 0–2), protože zajišťuje vysokou doručitelnost sdělení a zároveň umožňuje měřit míru zapojení adresátů. To je zásadní pro efektivní fungování obchodního oddělení.

Akvizitor v oblasti logistiky přitom plní roli jakéhosi „dirigenta“ rozšířeného B2B prodejního trychtýře. Klíčovým faktorem pro zrychlení celého obchodního cyklu a zvýšení výnosnosti společnosti LOG-IN CZ je efektivní automatizace právě těchto raných fází – od vyhledávání kontaktů až po jejich kvalifikaci.

1.1 Role umělé inteligence a RPA platforem v B2B prodeji

Umělá inteligence již dnes nachází pevné místo v B2B prodeji. Podle průzkumu Salesforce (2025) využívá 47 % obchodníků generativní AI k tvorbě e-mailů a zpráv určených k oslovování zákazníků, zatímco dalších 16 % spoléhá na AI při identifikaci potenciálních klientů. V logistickém sektoru byl globální trh s generativní AI v roce 2024 odhadnut na **1,3 miliardy USD (≈ 28 miliardy Kč)** a očekává se jeho růst na více než **7 miliard USD (≈ 150,7 miliardy Kč)** do roku 2030 (Mordor Intelligence, 2025). Významně roste také trh s robotickou automatizací procesů (RPA), který podle společnosti Technavio (2025) ročně expanduje o více než 40 % a do roku 2029 může překročit hodnotu **40 miliard USD (≈ 861 miliard Kč)**. Tyto

trendy ukazují, že automatizace prvních fází obchodního trychtýře se stává běžným standardem, nikoli výjimkou.

1.1.1 Typický „technologický stack“ digitálního akvizitora

Moderní akvizitor dnes nespoléhá pouze na svůj obchodní talent, ale i na sofistikované digitální nástroje. Umělá inteligence a RPA platformy spolupracují v různých fázích akvizičního procesu – od vyhledávání kontaktů až po analýzu výsledků. LLM modely, jako je ChatGPT, pomáhají s generováním vyhledávacích dotazů, zatímco scrapovací nástroje Apify nebo API parsery extrahují relevantní data. Validace e-mailových adres a scoring probíhají automaticky díky RPA robotům. Obsah e-mailů i zpráv na LinkedIn je personalizován pomocí AI, což zvyšuje míru prokliků (CTR) až o 30 % (Hubspot, 2025). Orchestrace kampaní a aktualizace CRM jsou řízeny nástroji jako Make.com, což eliminuje potřebu ruční práce. Takto sestavený „stack“ umožňuje akvizitorům reagovat rychleji a přesněji na tržní příležitosti.

Tabulka 2 Technologické komponenty digitálního akvizitora

Úkol	AI komponenta	RPA komponenta	Praktický přínos
Vyhledávání leadů (prospecting)	Large Language Models (LLM) modely (ChatGPT 3.5/4) generují vyhledávací dotazy a filtry	Scrapery Apify, API parsery (Google Maps, Europages)	10–100× zrychlení sběru kontaktů
Čištění a obohacení dat	Klasifikace odvětví, scoring, priorizace	Hromadná validace e-mailů pomocí Simple Mail Transfer Protocol (SMTP) robotů	Redukce „mrtvých“ adres, vyšší doručitelnost
Tvorba obsahu	Generování personalizovaných e-mailů, UTP, zpráv pro LinkedIn	-	20–30% nárůst CTR díky relevantnímu obsahu
Orchestrace kampaní	Machine Learning (ML, strojové učení) modely určují čas odeslání, pořadí interakcí	Make.com / Zapier koordinují úkoly, aktualizují CRM	Konzistentní multikanálová komunikace bez ruční práce
Analytika a optimalizace	AI atribuce leadů, prediktivní scoring	Automatický import metrik (open, click, reply)	Rychlé A/B testování, vyšší konverze na SQL

Zdroj: upraveno podle HubSpot (2024), Salesforce (2024), vlastní úprava autora

V případě **LOG-IN CZ** právě kombinace **ChatGPT + Make + Apify** pokrývá první dva sloupce tabulky:

- **LLM modely** generují vyhledávací dotazy;
- **Make scénáře** spouštějí scrapery, zpracovávají výsledky a zapisují je do **Google Sheets**.

1.1.2 Dopad na akviziční trychtýř

Vliv těchto technologií na tradiční obchodní cyklus je značný. Generativní modely dokážou připravit personalizovanou nabídku během několika sekund, čímž dochází ke zkrácení fáze od prvního kontaktu po kvalifikaci (fáze 0–2) o týdny. Umělá inteligence zároveň eliminuje nevhodné leady na základě objemu přepravy nebo lokality, čímž zvyšuje kvalitu potenciálních příležitostí. Personalizace obsahu pak probíhá dynamicky – AI například doplní aktuální trendy oboru, aniž by bylo nutné zapojit copywritera. Podle BusinessWire (2025) vzrostl podíl AI chatbotů využívaných k předkvalifikaci dotazů o 42 % v roce 2024.

1.1.3 Regulační a organizační omezení

V rámci nasazování umělé inteligence a robotické automatizace procesů (RPA) je nutné zohlednit několik klíčových aspektů. Z pohledu GDPR musí AI systémy vyloučit zpracování osobních e-mailových adres bez souhlasu a zároveň uchovávat záznamy o udělených souhlasech. Tyto požadavky jsou v praxi zajišťovány automatizovaně prostřednictvím RPA robotů, kteří aplikují příslušné filtry.

Důležitou roli hraje také hygiena dat. Používání nekvalitních nebo neověřených vstupních zdrojů vede k vysoké míře odmítnutých e-mailů, což negativně ovlivňuje reputaci odesílací domény. Z tohoto důvodu je klíčová předběžná validace adres prostřednictvím RPA a řízené zahřívání (warm-up) subdomény, jak je popsáno v části § 4.3.

Z pohledu organizačního řízení změn (change management) upozorňuje studie Boston Consulting Group (2024) na to, že 74 % firem má problém se škálováním AI pilotních projektů právě kvůli interním organizačním překážkám. Úspěšné zavádění AI systémů tedy vyžaduje nejen technickou připravenost, ale také školení zaměstnanců a jasné rozdělení odpovědností mezi automatizované nástroje („robota“) a lidské pracovníky.

1.1.4 Synergie AI a RPA: „digitální kopilot“ akvizitora

Spojením kognitivních schopností velkých jazykových modelů (LLM), jako je generování textu, klasifikace a vyvozování závěrů, s výkonností platforem pro robotickou automatizaci procesů (RPA), zahrnující shromažďování dat, zápis do CRM systémů a spouštění e-mailových kampaní, vzniká komplexní end-to-end proces. V tomto modelu se lidský pracovník může soustředit na strategii, analýzu a vyjednávání, zatímco rutinní a technicky opakovatelné úkoly jsou zcela automatizovány.

Pro společnost LOG-IN CZ tento přístup znamená praktické přínosy, jako je přístup k více než 6 000 validním e-mailovým adresám bez nutnosti navýšení počtu zaměstnanců,

dosažení 30% míry otevření e-mailů (open rate) a 5% míry prokliku (CTR) již v první iteraci kampaně, a zajištění transparentního, auditovatelného datového toku plně v souladu s nařízením GDPR.

Z výše uvedeného vyplývá, že nástroje umělé inteligence a RPA již nejsou pouze prostředkem k urychlení práce, ale tvoří základní stavební kámen nového digitálního standardu v oblasti B2B prodeje - standardu, který představuje klíčový konkurenční faktor zejména pro středně velké logistické společnosti.

1.2 Přehled technologií použitých v projektu

Jak je patrné z tabulky 3, jednotlivé komponenty se vzájemně doplňují a vytvářejí plynulý automatizovaný tok. Právě taková kombinace je klíčem k rychlému a udržitelnému získávání nových klientů.

Tabulka 3 Přehled technologií využitých v projektu LOG-IN CZ

Komponenta	Stručná charakteristika	Role v projektu LOG-IN.CZ
ChatGPT 3.5 / 4 (OpenAI)	Rodina velkých jazykových modelů. Verze Generative Pre-trained Transformer (GPT)-4.1 (API) podporuje až 1 milion tokenů kontextu a vykazuje vyšší přesnost při generování a kódování. Nejnovější řada o-modelů (o3, o4-mini) je optimalizována pro rychlost a aplikační úkoly.	1) Generování vyhledávacích dotazů (příkazy /maps, /search). 2) Vytváření UTP a odpovědí klientům (/utp, /reply). 3) Klasifikace odvětví, AI-skórování leadů.
Make.com (ex-Integromat)	No-code platforma pro vizuální automatizaci: 200+ připravených integrací, AI-agenti, vestavěná kompatibilita s GDPR a System and Organization Controls 2 (SOC 2).	Orchestrace celého procesu: spouštění Apify scraperů, zápis dat do Google Sheets, spuštění rozesílek, synchronizace s Telegram botem a CRM.
Apify	Cloudové prostředí pro tvorbu a spuštění „actorů“ – scraperových scénářů. Připravený Google Maps Scraper získává názvy firem, telefony, Uniform Resource Locator (URL), geo souřadnice a další informace podle klíče „kategorie + lokalita“.	Rychlý sběr „dlouhého seznamu“ (fáze 0 v akvizičním trychtýři): až 1000 organizací na jeden dotaz; rozšiřitelné na Europages, Firmy CZ a další katalogy.
SMTP infrastruktura	Samostatná subdoména mail.login.cz na platformě Google Workspace; konfigurace Domain Name System (DNS) záznamů SPF (Sender Policy Framework), DKIM (Domain Keys Identified Mail) a DMARC (Domain-based Message Authentication, Reporting and Conformance). Postupy zahrnují zahřívání domény (domain warm-up), validaci e-mailových adres a dynamické tempo odesílání zpráv.	Hromadná rozesílka (až 300 e-mailů/den v pilotní fázi), sledování open-/click-metrik, předání „horkých“ leadů obchodnímu oddělení.

Zdroj: Autor podle OpenAI (2024), Make.com (2024), Apify (2024), Google Workspace (2024)

Integrace technologií v rámci akvizičního systému společnosti LOG-IN CZ byla postavena na několika klíčových principech. Jedním z nich je flexibilní správa verzí velkých jazykových modelů (LLM) – pro běžné návrhy e-mailů byl využíván model GPT-3.5 z důvodu nižších nákladů, zatímco pro tvorbu personalizovaných unikátních prodejních nabídek (UTP) a dlouhých HTML e-mailů byl použit model GPT-4.1, který umožňuje práci s širším kontextem.

Dalším důležitým prvkem jsou end-to-end procesy bez nutnosti programování. Platforma Make.com propojuje Telegram, Apify, Google Sheets a Gmail API do jednotného pracovního prostředí, ve kterém akvizitor pracuje s přehledným vizuálním rozhraním. Veškerá logika systému je uložena v podobě vizuálních scénářů.

Součástí řešení je také škálovatelný scraping – Apify actory běží paralelně a limit Google Maps API je obcházen rozložením dotazů mezi více proxy pooly. To umožňuje stáhnout až 10 000 záznamů denně dle potřeby. Kromě toho je pod pečlivou kontrolou také reputace e-mailové komunikace. Subdoména je zahřívána po dobu sedmi dní podle schématu 25–50–100 e-mailů denně a nedoručitelné adresy jsou automaticky označovány v Google Sheets pomocí webhooku, který zpracovává hard-bounce odpovědi.

V souladu s principem „GDPR by design“ platforma Make.com uchovává detailní logy všech operací a RPA filtr automaticky vylučuje osobní e-mailové adresy (např. peter.novak@..), přičemž ponechává pouze obecné schránky (např. info@, sales@).

Tento technologický stack umožnil společnosti LOG-IN CZ vytvořit nepřetržitý a měřitelný tok validovaných obchodních leadů, a to bez nutnosti rozšiřovat tým a s minimálními náklady na technickou infrastrukturu.

1.3 Právní rámec (GDPR a e-mailový marketing)

Hlavní požadavky na elektronický B2B outreach v Evropské unii vycházejí z kombinace dvou právních předpisů – obecného nařízení o ochraně osobních údajů (GDPR) a směrnice ePrivacy. Tyto normy společně určují, za jakých podmínek lze oslovovat firemní kontakty e-mailem, jaké informace je možné zpracovávat a jak musí být nastavena možnost odhlášení z komunikace. Shrnutí klíčových ustanovení obou předpisů a jejich praktického dopadu na realizaci B2B kampaní uvádí následující tabulka.

Tabulka 4 Legislativní rámec B2B e-mailingu podle GDPR a ePrivacy

Předpis	Klíčová ustanovení pro marketing	Praktický význam
GDPR (Nařízení EU 2016/679)	• Šest zákonných důvodů pro zpracování údajů (čl. 6); pro přímý B2B marketing se nejčastěji používá oprávněný zájem - čl. 6 (1)(f), potvrzený Recitalem 47. • Právo příjemce vznést námitku (čl. 21 (2)); povinnost poskytnout snadné odhlášení (čl. 7 (3)). • Zásady minimalizace, přesnosti, omezeného uchovávání (čl. 5).	Odesílatel musí provést test rovnováhy zájmů a uchovávat jeho výsledky; použití osobních adres bez souhlasu je zakázáno.
Směrnice ePrivacy 2002/58/ES (a její česká transpozice, zákon č. 480/2004 Sb.)	Zakazuje „nevyžádaná obchodní sdělení“ bez předchozího souhlasu, ale připouští výjimku s možností odhlášení pro B2B adresy, pokud byl kontakt získán „v souvislosti s prodejem produktu / služby“ a e-mail se týká „podobných“ služeb.	V Česku Úřadu pro ochranu osobních údajů (ÚOOÚ) interpretuje obecné e-mailové schránky (např. info@, sales@) jako méně citlivé: lze se odvolat na oprávněný zájem; osobní adresy vyžadují souhlas.

Zdroj: Autor podle EUROPEAN COMMISSION (2024), EDPB (2024), ÚOOÚ (2024)

Za porušení pravidel GDPR hrozí pokuty až do výše 20 milionů € nebo 4 % celosvětového obratu; ke konci roku 2024 již byly uvaleny sankce na velké logistické společnosti za neautorizované rozesílky.

V říjnu 2024 zveřejnil Evropský sbor pro ochranu osobních údajů (EDPB) **nové Pokyny 1/2024**, které se detailněji zabývají využíváním oprávněného zájmu v rámci přímého marketingu. Podle těchto pokynů musí být zájem **konkrétní a oprávněný**, přičemž je třeba dodržet principy proporcionality a respektovat oprávněná očekávání příjemce.

Evropský sbor pro ochranu osobních údajů (EDPB) doporučuje organizacím používat jednoduchý tříkrokový postup pro posouzení zákonnosti zpracování osobních údajů na základě oprávněného zájmu. Tento přístup zahrnuje: (1) identifikaci oprávněného zájmu, (2) posouzení nezbytnosti zpracování a (3) vyvážení tohoto zájmu s právy a svobodami subjektu údajů.

Pro společnost LOG-IN CZ to v praxi znamená, že je vhodné uchovávat záznamy o vyhodnocení rovnováhy zájmů ve firemní dokumentaci, zpracovávat pouze nezbytné údaje,

jako je pracovní e-mail a pozice adresáta, a zároveň zajistit jednoduchou a jasnou možnost odhlášení z další komunikace. To může být realizováno například prostřednictvím odkazu „unsubscribe“ nebo výzvy „odpovědět STOP“.

V souvislosti s provozem automatizovaných e-mailových kampaní je nezbytné identifikovat a řízeně minimalizovat související rizika, která mohou mít dopad jak na technickou stránku rozesílky, tak na právní soulad se směrnicemi GDPR a ePrivacy. Následující tabulka shrnuje hlavní rizika spojená s rozesíláním zpráv přes SMTP protokol a zároveň uvádí konkrétní mitigace, které byly implementovány v projektu LOG-IN CZ.

Tabulka 5 Rizika SMTP rozesílek a jejich mitigace v projektu LOG-IN CZ

Riziko	Mitigační opatření realizované v projektu
Hard-bounce / spam-trap → pokles reputace domény a blokace	Skriptová validace SMTP adres, vyloučení catch-all bez potvrzeného Mail Exchange (MX); zahřívání subdomény mail.login.cz po dobu 7 dní (25–50–100 e-mailů denně).
Zpracování osobních údajů třetími stranami (OpenAI, Make, Apify)	Smlouvy o zpracování údajů (Make má certifikaci SOC 2-Type II, Apify - ISO 27001); data jsou přenášena přes Transport Layer Security (TLS) a logována v regionu EU.
Nedostatek transparentnosti	V každém e-mailu: právní název LOG-IN.CZ, kontakt na Data Protection Officer (DPO), odkaz na Zásady ochrany osobních údajů, unikátní Identifikátor (ID) kampaně.
Právo na výmaz / námitku	V databázi Google Sheets příznak „opt-out“; scénář v Make okamžitě zastaví veškerý další kontakt.

Zdroj: Autor podle OpenAI (2025), Make.com (2024), Apify (2024), Google Workspace (2024)

V rámci dodržování zásad GDPR je v systému implementován princip „privacy by design“. Filtr v Make vylučuje osobní e-mailové adresy a prompt velkého jazykového modelu obsahuje pravidlo, že nesmí být používány osobní údaje kromě těch, které jsou veřejně dostupné ve vizitce. Přenosy dat mezi systémy probíhají v souladu s evropskou legislativou, přičemž OpenAI zpracovává data na serverech umístěných v EU a standardní smluvní doložky (Standard Contractual Clauses) jsou aktivovány ve výchozím nastavení. Použití ChatGPT slouží výhradně k tvorbě textu a nevede k automatizovanému rozhodování s právním účinkem, takže se neuplatňuje článek 22 GDPR o rozhodnutích bez lidského zásahu. Pro zajištění transparentnosti a možnosti auditu ze strany Úřadu pro ochranu osobních údajů (ÚOOÚ) systémy Apify a Make zaznamenávají trasování požadavků včetně IP adres, časových razítek a objemu přenesených dat.

Aby bylo možné garantovat soulad akvizičního systému s požadavky na ochranu osobních údajů, je nezbytné vytvořit systematický kontrolní rámec pokrývající všechny klíčové oblasti zpracování dat. V projektu LOG-IN CZ byl proto sestaven tzv. compliance checklist,

který mapuje jednotlivé právní a technické požadavky, a zároveň ověřuje jejich naplnění v praxi. Následující tabulka shrnuje hlavní kontrolní body a stav jejich implementace.

Tabulka 6 Compliance checklist projektu LOG-IN CZ

Kontrolní bod	Status
Rovnováha mezi oprávněným zájmem a očekáváním soukromí	Zdokumentováno (formuláře LIA) **Příloha H**
Smlouvy o zpracování údajů s externími zpracovateli	Podepsány (OpenAI, Google, Apify, Make)
SPF / DKIM / DMARC na subdoméně	Nastaveno
Opt-out mechanismus ≤ 2 kliky	Implementováno
Retence: mazání neaktivních leadů ≥ 12 měsíců	Ve skriptu pro automatické čištění
Evidence odhlášení (unsubscribe log)	Google Sheets + webhook

Zdroj: autor

Tímto způsobem je projekt postaven v plném souladu s aktuálním evropským a českým regulačním rámcem: zpracovávají se pouze nezbytně nutné údaje, rozesílka se zakládá na „oprávněném zájmu“ u obecných e-mailových schránek a automatizační infrastruktura podporuje princip odpovědnosti. To snižuje regulatorní rizika a zajišťuje udržitelnou reputaci domény, což má přímý dopad na doručitelnost a efektivitu akvizice.

2 ANALÝZA VÝCHOZÍHO STAVU LOG-IN.CZ

Kořeny společnosti LOG-IN sahají do roku 1984, kdy byla v Německu založena firma specializující se na chráněnou přepravu cenných nákladů. V roce 2006 bylo založeno české právnické zastoupení - LOG-IN CZ s.r.o. (Pardubice, Doubravice 106), které je dnes součástí dopravně-logistického holdingu MATTELI Group a.s.

Společnost LOG-IN vystupuje pod heslem „Security for your cargo“ a specializuje se na mezinárodní přepravu vysoce hodnotných produktů, jako jsou elektronika, tabákové výrobky a další zboží s vysokou přidanou hodnotou, a to v rámci celé Evropy.

Mezi hlavní služby patří zejména využití specializovaných vozidel, která splňují nejpřísnější bezpečnostní normy dle TAPA TSR úrovně 1 (Transported Asset Protection Association – Trucking Security Requirements) a BAT V1 (Business Approval Tool). Dále nabízí teplotně kontrolovanou přepravu v režimu FTL (Full Truck Load) a LTL (Less Than Truck Load), doplněnou o možnost doprovodu nákladu, který je řízen prostřednictvím vlastního bezpečnostního centra s nepřetržitým provozem 24/7. Samozřejmostí je také multimodální logistika, která zahrnuje efektivní napojení na subdodavatelské dopravce.

Společnost disponuje flotilou cca 400 vysoce zabezpečených tahačů a návěsů (tautliner, box, frigo, jumbo), všechny v emisní třídě EURO 5–6, a navíc využívá solo nákladní vozy a dodávky partnerů. Mezi potvrzení kvality patří certifikace ISO 9001:2015 (mezinárodní norma systému řízení kvality), AEO (Authorised Economic Operator – oprávněný hospodářský subjekt) a hodnocení Dun & Bradstreet „AAA“ (nejvyšší úroveň důvěryhodnosti firmy podle mezinárodní ratingové agentury)

Společnost zaměstnává přibližně 173 osob, struktura se neformálně dělí na pět klíčových bloků:

Organizační struktura společnosti LOG-IN CZ vychází z centralizovaného modelu, jehož vrcholem je **generální ředitel**. Ten přímo dohlíží na čtyři hlavní organizační větve, které společně zajišťují efektivní řízení každodenního provozu, bezpečnost, obchodní růst i podpůrné funkce. Struktura je navržena s důrazem na funkčnost, dohled a specializaci.

Tabulka 7 Klíčové větve organizační struktury společnosti

Větev	Podřízené jednotky	Hlavní funkce
1. Provozní větev (Operations)	1. Dispečeři 2. Řidiči	Plánování a řízení přeprav, přidělování jízd, údržba vozidel, sledování tras
2. Obchodní větev (Sales & Acquisition)	–	Vyhledávání a akvizice nových klientů, správa obchodních nabídek a tendrů
3. Bezpečnostní větev (Security Control Centre)	–	24/7 monitoring, řízení incidentů, zajištění souladu s bezpečnostními normami (např. TAPA TSR)
4. Podpůrná (outsourcovaná) větev	1. Finance 2. Technické zajištění 3. Projektoví manažeři skupiny	Sdílené služby v rámci holdingu (např. účetnictví, technická podpora, reporting, vedení IT projektů)

Zdroj: Autor podle veřejné firemní informace (LOG-IN CZ, 2024)

Díky kombinaci moderních bezpečnostních technologií, rozsáhlého vozového parku a prestižních certifikací si LOG-IN CZ buduje silné postavení v segmentu přepravy cenného zboží. Namísto cenové konkurence firma sází na spolehlivost, transparentnost a odpovědnost. Právě v této oblasti může automatizace akvizice přinést největší efektivitu.

2.1 Stávající procesy získávání zákazníků

Před zahájením automatizace neměla společnost LOG-IN CZ s.r.o. interního akvizitora. Vyhledávání nových zákazníků („lov“) bylo rozděleno mezi dva obchodníky oddělení Sales, kteří současně řešili smluvní záležitosti, tarifní nabídky a částečně i správu stávajících klientů.

Proces získávání obchodních kontaktů (lead generation) ve společnosti LOG-IN CZ probíhal prostřednictvím několika základních kanálů. Přibližně 30 % nových poptávek pocházelo z doporučení od stávajících partnerů a spedičních společností. Dalším zdrojem byly pravidelné účasti ve výběrových řízeních velkých výrobců elektroniky, které se konaly zhruba jednou až dvakrát za čtvrtletí. Společnost rovněž využívala občasné rozesílky „studených“ e-mailů, odesílaných z obecné firemní e-mailové schránky, a kontakty byly manuálně vyhledávány na platformách LinkedIn a Facebook, včetně individuální komunikace s potenciálními klienty.

Používané nástroje byly převážně základního charakteru - primárně Microsoft Outlook pro e-mailovou komunikaci a Excelový soubor s názvem „Seznam kontaktů“, uložený na lokálním firemním serveru. Telefonní hovory obchodníků byly zaznamenávány výhradně do jejich osobních poznámkových bloků.

CRM systém ve firmě existuje, avšak není využíván v plném rozsahu. Slouží spíše jako pasivní úložiště základních údajů o firmách a přehledu uskutečněných obchodních případů, bez aktivního zapojení do každodenní akviziční činnosti nebo sledování interakcí se zákazníky.

Komunikační šablony byly zastoupeny jedním e-mailovým skriptem v češtině a angličtině o délce přibližně 180 slov, bez jakékoli dynamické personalizace.

Žádné oficiální metriky (např. open rate, click rate) se nesledovaly. Při pilotní kampani v září 2024 bylo odesláno 115 e-mailů, z nichž přibližně 50 % skončilo ve spamu a nebyla zaznamenána žádná odpověď.

Z hlediska souladu s GDPR nebyla provedena segmentace adres - do rozesílky se mohly dostat i osobní e-maily a logy souhlasů nebyly ukládány, což představovalo regulační riziko.

Analýza dosavadního přístupu k akvizičním aktivitám ve společnosti LOG-IN CZ odhalila několik zásadních problémů, které negativně ovlivňovaly efektivitu a soulad s legislativními požadavky. Prvním z nich byl nedostatek kapacit – obchodníci, kterých bylo pouze dva, trávili až 60 % své pracovní doby rutinními činnostmi, jako je manuální vyhledávání e-mailových kontaktů, kopírování údajů a ruční rozesílání zpráv.

Systém postrádal transparentnost, neboť ačkoliv společnost disponuje CRM nástrojem, ten není využíván v plné míře. Pokrok u jednotlivých leadů byl proto často vyhodnocován subjektivně a bez jasně definovaných metrik. Z technického hlediska byla doména login.cz nastavena pouze se základním SPF záznamem, přičemž chyběly klíčové autentizační protokoly DKIM a DMARC, což výrazně snižovalo doručitelnost e-mailových zpráv.

Další slabinou byla absence systému pro validaci a „zahřívání“ databáze kontaktů – vysoký podíl neplatných adres vedl k nárůstu bounce-rate a ohrožení reputace domény. Neprobíhala ani segmentace adresátů, což mohlo vést k rozesílce na osobní e-maily bez předchozího souhlasu. V systému navíc chyběly záznamy o udělených souhlasech, což představovalo vážné regulační riziko v kontextu GDPR.

2.2 Stávající procesy udržení zákazníků

Ve společnosti LOG-IN CZ nebyl po podpisu smlouvy klientům přidělován samostatný account manažer – péči o zákazníky zajišťoval dispečer z oddělení Operations. Komunikace probíhala prostřednictvím telefonu, firemního WhatsAppu a e-mailu, ale chyběly jakékoliv self-service portály či integrace klienta s back-officem přes EDI nebo CRM, s výjimkou sledování dopravy pomocí GPS. Kontrola spokojenosti zákazníků se omezovala na občasné

telefonické dotazy typu „jak proběhla poslední přeprava“, formální průzkumy jako Net Promoter Score (NPS) se neprováděly a reklamace se evidovaly ručně v Excelu. Retenční metriky, například Customer Lifetime Value (CLV) nebo churn-rate, se systematicky nesledovaly.

Analýza navíc ukázala, že i klienti s vysokým potenciálem často nedostávají proaktivní nabídky, protože oddělení Sales je nadměrně vytížené vyhledáváním nových leadů. Chybí také formálně nastavené KPI pro udržení stávajících zákazníků, což vede k tomu, že iniciativy na zlepšení služeb, jako jsou EDI nebo SLA dashboardy, jsou realizovány spíše ad-hoc. Přestože společnost LOG-IN CZ vykazuje vysokou kvalitu v oblasti samotné přepravy a operativy, akviziční a retenční procesy byly dosud řízeny převážně manuálně a bez využití moderních nástrojů. Omezené využívání CRM systému, nejasně definované metriky a slabá technická infrastruktura představují zásadní překážky pro další růst. Tyto skutečnosti se staly výchozím bodem pro návrh automatizovaného systému, jehož cílem je tyto nedostatky odstranit a přinést společnosti dlouhodobou konkurenční výhodu.

2.3 Identifikované problémy a cílové KPI projektu

Aby bylo možné systémově zaměřit investice do automatizace, byla provedena analýza kořenových příčin nízké akviziční výkonnosti.

Tabulka 8 Přehled klíčových problémů a jejich řešení v rámci projektu

№	Problém	Dopady na podnikání	Řešení v rámci projektu
1	Absence interního akvizitora, rutinní úkoly jsou rozděleny mezi 2 manažery	Pomalý růst zákaznické báze; vysoká časová náročnost	Telegram bot + scénáře v Make automatizují vyhledávání a primární kvalifikaci
2	Ruční sběr e-mailových kontaktů a absence validace	115 adres → ≈ 50 % e-mailů ve spamu, bounce-rate se neměřili	SMTP validační skript, catch-all filtr, vytvoření subdomény a zahřívání
3	Jednotná šablona bez personalizace	Nulová odezva (115 e-mailů → 0 odpovědí)	AI generování předmětů a UTP, A/B testy, dynamické bloky v HTML
4	Chybí transparentní analytika	Vedení nevidí akviziční funel, nelze optimalizovat proces	Google Sheets dashboardy a týdenní reporty
5	Pouze SPF, bez DKIM/DMARC; nerespektování GDPR omezení	Nízká reputace domény; regulatorní rizika	Kompletní e-mailový stack (SPF + DKIM + DMARC), LIA dokumentace, opt-out
6	Reaktivní přístup k udržení zákazníků, chybí formální NPS	Nevyužitý upsell potenciál, churn se nesleduje	Přenos „horkých“ leadů do Sales + plán zavedení NPS průzkumů.

Zdroj: Upraveno autorem podle interní analýza procesů LOG-IN CZ s.r.o. (2024)

V rámci projektu byly stanoveny čtyři klíčové metriky, které slouží k průběžnému sledování výkonnosti automatizovaného akvizičního systému. První z nich je kvalita databáze, která tvoří základ všech dalších ukazatelů. Je měřena denně pomocí validačního skriptu, který odhaluje neplatné nebo neaktivní e-mailové adresy. Druhým ukazatelem je míra otevření e-mailů (open rate) a míra prokliků (CTR), jež reflektují relevanci a atraktivitu obsahu. Tyto hodnoty jsou pravidelně optimalizovány prostřednictvím A/B testování předmětů zpráv a šablon, které probíhá každé dva týdny.

Dalším klíčovým ukazatelem je poměr mezi kvalifikovanými leady (SQL) a skutečně uzavřenými kontrakty. Tento integrační KPI hodnotí, jak dobře funguje provázanost automatizovaného systému s lidskou obchodní činností. Poslední sledovanou oblastí je dodržování GDPR, kde projekt uplatňuje politiku nulové tolerance – jakákoli stížnost automaticky blokuje další kontakt s danou adresou.

Cílem je dosáhnout téměř ideální doručitelnosti e-mailů (≈ 95 %), výrazně překonat odvětvový průměr míry otevření (open rate), zajistit stabilní tok kvalifikovaných leadů a současně snížit administrativní zátěž obchodního oddělení při plném respektování regulatorních požadavků.

3 NÁVRH AUTOMATIZOVANÉHO AKVIZIČNÍHO SYSTÉMU

Aby nový systém automatizace přinášel reálný přínos pro obchodní výkon společnosti LOG-IN CZ, bylo nutné již od počátku jasně definovat, co přesně má umět a jaké podmínky musí splnit. Tyto požadavky vycházely z reálné praxe firmy a byly rozděleny do tří hlavních kategorií: funkční, nefunkční a regulační.

Pro prioritizaci jednotlivých bodů byla využita metoda MoSCoW (Must, Should, Could, Won't) (STAPLETON, 1997), která umožnila stanovit, které požadavky jsou pro funkčnost systému naprosto klíčové a které lze případně řešit až v pozdější fázi.

- **M - Must have**
  Musí být – Zcela nezbytné požadavky. Bez jejich splnění by projekt nemohl fungovat nebo by ztratil smysl.
- **S - Should have**
  Mělo by být – Důležité, ale ne kritické požadavky. Je vhodné je realizovat, ale v případě omezených zdrojů je možné je odložit.
- **C - Could have**
  Mohlo by být – Doplňkové funkce nebo vylepšení. Nejsou prioritní, ale pokud bude čas a rozpočet, mohou být zahrnuty.
- **W - Won't have (this time)**
  Nebude (tentokrát) – Požadavky, které v tomto vydání nebudou zahrnuty. Mohou být zvažovány v budoucnu.

Tato metoda pomáhá stanovit priority a řídit očekávání všech zúčastněných stran v projektu.

Aby bylo možné efektivně nasadit automatizovaný akviziční systém ve společnosti LOG-IN CZ, bylo nezbytné stanovit soubor konkrétních funkčních požadavků a k nim odpovídajících akceptačních kritérií. Tyto požadavky byly rozděleny podle priority a pokrývají všechny klíčové fáze akvizičního procesu - od vyhledávání kontaktů přes validaci a personalizaci až po sledování výsledků a předávání kvalifikovaných leadů obchodnímu týmu. Přehled všech požadavků uvádí následující tabulka 9.

Tabulka 9 Funkční požadavky a akceptační kritéria systému automatizace akvizice

ID	Požadavek	Popis / akceptační kritérium	Priorita
F-1	Hromadné vyhledávání leadů	Systém musí shromažďovat ≥ 1 000 nových firem/den z Google Maps, Europages, Firmy CZ, LinkedIn, Facebook.	Must
F-2	Validace e-mailů	SMTP ping, filtr catch-all, vyloučení „šedých“ a osobních adres; bounce ≤ 2 %.	Must
F-3	AI scoring leadů	LLM model přiřazuje hodnocení (High/Medium/Low) na základě odvětví, obratu, tras.	Should
F-4	Personalizovaný e-mail	Generování předmětu a těla (HTML / Plain) v jazyce adresáta (RU/CZ/EN); doplnění UTP a názvu firmy.	Must
F-5	Multikanálový outreach	Telegram bot spouští /maps, /search, /europages; Make scénáře spouští rozesílku e-mailů a zpráv na LinkedIn/Facebook.	Must
F-6	Sledování aktivity	Záznam open, click, reply; aktualizace polí “Last Touch”, “Status” v Google Sheets; týdenní dashboard.	Must
F-7	Automatický opt-out	Odkaz na odhlášení a „Odpověď STOP“ přidají e-mail na blacklist; další kontakt je zakázán.	Must
F-8	Předání „horkých“ leadů	Při kliknutí NEBO odpovědi se vytvoří úkol „Callback“ pro obchodníka na email	Must
F-9	A/B testování	Paralelní rozesílání ≥ 2 variant předmětu/CTA; systém ukládá statistiky a označuje vítěze.	Could
F-10	Záznam aktivit	Logy API požadavků, změn v databázi a rozesílek se uchovávají ≥ 12 měsíců pro audit.	Must

Zdroj: autor

Mezi **nefunkční požadavky** automatizovaného systému akvizice patřila celá řada kritérií, podle kterých byl návrh hodnocen. V oblasti výkonu systém umožňoval zpracování až 1 000 leadů během čtyř hodin a rozesílání až 300 e-mailů denně v pilotní fázi, s možností dalšího škálování až na 1 500 zpráv denně. Z hlediska dostupnosti byla zajištěna nepřetržitá funkčnost platformy Make i napojeného Telegram bota, bez zaznamenaných výpadků.

Bezpečnostní aspekty byly řešeny pomocí šifrování TLS ve verzi 1.2 a vyšší, pravidelnou rotací API klíčů každých 90 dní a zavedením dvoufaktorové autentizace v rámci Google Workspace. Důraz byl kladen i na škálovatelnost – systém umožňoval bez zásahu do základního jádra připojovat nové zdroje dat, například Kompass nebo ThomasNet.

Co se týče uživatelské přívětivosti, obchodníkům stačilo dvouhodinové zaškolení pro práci s chatbotem, přičemž veškerá důležitá data byla soustředěna do přehledné tabulky v Google Sheets. Nákladová efektivita systému byla rovněž pozitivní – měsíční náklady na API a hosting nepřesahovaly 540 Kč v pilotní verzi a jednotková cena za jeden validní lead se pohybovala pod hranicí 10 Kč.

Plnění regulačních a právních požadavků v oblasti e-mailové komunikace vyžaduje dodržování několika klíčových pravidel. V souladu s nařízením GDPR probíhá zpracování

údajů na základě oprávněného zájmu, přičemž jsou vyloučeny osobní e-mailové adresy, pokud není udělen souhlas. Zároveň je zajištěna možnost jednoduchého odhlášení z odběru obchodních sdělení, a to maximálně na dva kliky, a je vedena evidence souhlasů a záznamů o zpracování.

Podle směrnice ePrivacy a zákona č. 480/2004 Sb. musí být každý odesílatel jasně identifikovatelný a zpráva musí být zřetelně označena jako „obchodní sdělení“.

Z hlediska reputace e-mailové komunikace je klíčové správné nastavení autentizačních záznamů SPF, DKIM a DMARC. Kromě toho se provádí tzv. zahřívání subdomény po dobu sedmi dnů a aktivně se sleduje podíl stížností na spam, který nesmí překročit hodnotu 0,3 %.

Pro efektivní fungování navrženého systému automatizace akvizice bylo důležité jasně definovat jednotlivé uživatelské role a jejich odpovědnosti. Každý uživatel či komponenta systému má specifickou úlohu v rámci celého procesu - od technické správy až po přímou interakci s klienty. Následující tabulka 10 přehledně shrnuje hlavní činnosti jednotlivých rolí zapojených do provozu systému.

Tabulka 10 Uživatelské role a jejich hlavní činnosti v navrženém systému

Role	Hlavní činnosti
Akviziční bot (Make / Telegram)	Provádí vyhledávání, scraping, validaci, rozesílku.
Obchodní manažer	Spouští příkazy bota, prohlíží „horké“ leady, volá klientům.
Informační technologie (IT) administrátor / Patron IT	Nastavuje DNS, sleduje SMTP logy, spravuje API klíče.
DPO / Compliance	Provádí audit logů, vyřizuje žádosti o výmaz údajů.

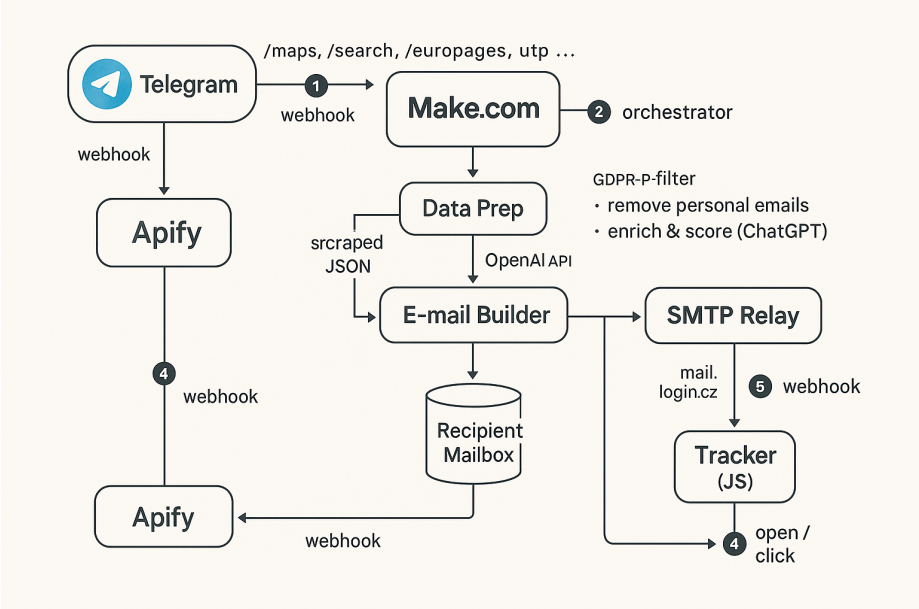
Zdroj: autor

Jako základ pro vyhodnocení úspěšnosti implementace akvizičního systému bylo stanoveno konkrétní kritérium „hotovo“, které definuje, kdy lze projekt považovat za dokončený a funkční v souladu s očekáváním. Systém se považuje za implementovaný, pokud je splněno několik měřitelných podmínek: alespoň 95 % e-mailů je doručeno do schránky příjemce, přičemž míra nedoručení (bounce rate) nepřesahuje 2 %. Po čtyřech týdnech testování musí databáze obsahovat minimálně 6 000 validních e-mailových adres. Datové dashboardy v Google Sheets se aktualizují automaticky alespoň jednou denně a obchodnímu týmu bylo předáno nejméně 50 kvalifikovaných leadů (SQL), z nichž alespoň 10 bylo telefonicky uzavřeno. Zároveň musí být ověřeno, že systém je v souladu s GDPR - tento soulad potvrdí pověřenec pro ochranu osobních údajů (DPO) a opt-out mechanismus musí být plně funkční a otestovaný.

Díky takto přesně formulovaným požadavkům vznikl jasně strukturovaný technický rámec celého projektu. Každý bod má definovaná akceptační kritéria, která umožňují v dalších fázích projektu objektivně posoudit, zda byly cíle naplněny. Tento přístup zároveň zajišťuje jednotné porozumění napříč týmem – od obchodníků až po správce technické infrastruktury.

3.1 Architektura řešení a výběr technologického stacku

Aby bylo možné lépe pochopit strukturu a vzájemné propojení jednotlivých nástrojů využitých v rámci automatizovaného akvizičního systému, následující schéma znázorňuje logický tok dat a operací mezi komponentami. Diagram popisuje celý proces od zadání vstupních parametrů až po doručení e-mailu a sledování jeho interakce. Zároveň ukazuje, jak byly propojeny služby jako Make, Apify, ChatGPT, e-mailová brána a nástroje pro sledování chování uživatele.



Obrázek 1 Logické schéma (end-to-end tok) (autor, generováno pomocí chat GPT)

Vysvětlení jednotlivých fází:

1. Telegram bot přijímá příkaz od akvizitora; parametry (kategorie, geolokace, limit, jazyk) se předávají do Make.
2. Make.com volá příslušné „Actors“ v Apify (Google Maps Scraper, Europages Scraper **(Příloha C)** atd.), sbírá surová data, spouští čištění a obohacení (Data Prep).
3. E-mail Builder pomocí ChatGPT generuje personalizovaný předmět, tělo, dynamické bloky. A/B logika: modul náhodně přiřazuje 50 %/50 % „Varianta A“ a „Varianta B“ a zapisuje toto pole do databáze.

4. SMTP Relay na subdoméně mail.login.cz (Google Workspace) odesílá e-maily přes Gmail SMTP; kompletní sada SPF + DKIM + DMARC je sladěna s „mateřskou“ doménou.
5. Tracker (1×1 px Web-Beacon + UTM parametr v CTA odkazu) spouští webhook Make, který aktualizuje stav „Opened“, „Clicked“.

3.1.1 Komponenty a odůvodnění výběru

Technologická architektura systému byla navržena s ohledem na rychlou implementaci, nízké provozní náklady, kompatibilitu s evropskou legislativou a uživatelskou přívětivost pro neprogramátorské role. Kombinace low-code nástrojů, open-source knihoven a cloudových služeb umožňuje snadnou údržbu, flexibilní rozšiřování a zároveň zachování plné kontroly nad daty. Použité technologie jsou rozděleny do jednotlivých vrstev podle jejich funkce v systému a každá z nich byla vybrána na základě jasně definovaných kritérií.

Tabulka 11 Použité technologie a jejich zdůvodnění ve vrstvách systému

Vrstva	Technologie	Důvod výběru
Uživatelské rozhraní pro akvizitora	Telegram Bot (python-telegram-bot)	minimální vstupní bariéra pro obchodníky; není třeba samostatné Graphical User Interface (GUI); push notifikace.
Orchestrace	Make.com (no-code)	vizuální scénáře, 200+ integrací, hosting v EU, vestavěné GDPR logování; rychlé úpravy bez nutnosti nasazení kódu.
Scraping	Apify Actors	připravené šablony pro Google Maps / LinkedIn / Facebook; horizontální škálování; integrovaný proxy management.
Obohacení a generování	ChatGPT 3.5 / 4 (OpenAI API)	vysoká jazyková variabilita (CZ / EN / RU); rychlý prototyping; versioning (levný 3.5 pro koncepty, 4 pro detailní UTP).
Datové úložiště	Google Sheets + Drive	dostatečná kapacita (10 mil. řádků), okamžité sdílení s vedením, import do Business Intelligence (BI); zdarma v rámci Google-Workspace.
Tracking	vlastní JS pixel (JavaScript pixel) a UTM parametry (Urchin Tracking Module – značení URL pro analytiku kampaní)	nezávislé na cizích cenících; data zůstávají v EU; flexibilní integrace do Make.
Monitoring / BI	App-script dashboard v Google Data Studio (Looker)	vizualizace KPI pro Chief Executive Officer (CEO), Human Resources (HR) a Sales bez dalších licencí.
Bezpečnost	Two-Factor Authentication (2FA) v Google Workspace	centrální správa uživatelů; shoda s ISO 27001.

Zdroj: autor

3.1.2 Rozšiřitelnost

Přidání nového zdroje leadů je možné jednoduchým způsobem – vytvořením nového Actoru v Apify nebo připojením externího API (například Kompass) a jeho napojením na existující router ve scénářích Make.

Navýšení objemu rozesílky je řešeno lineárně: použitím druhé subdomény a spuštěním paralelního Make scénáře.

Integrace CRM je připravena – současná struktura v Google Sheets obsahuje základní prvky, které lze v budoucnu rozšířit pomocí REST-push do systémů jako HubSpot nebo Pipedrive prostřednictvím standardního Make modulu.

Zvolený technologický stack je založen výhradně na cloudových službách v rámci jurisdikce EU. Díky tomu je možné výrazně minimalizovat množství vlastního kódu – přibližně 90 % logiky je zajištěno pomocí vizuálních scénářů v Make. Toto řešení umožňuje velmi rychlou iteraci, snadné škálování a zároveň splňuje všechny prioritní požadavky uvedené v části § 3

Při návrhu celého systému bylo nutné zohlednit také provozní správu a zabezpečení. Vzhledem k tomu, že systém pracuje s osobními údaji a zároveň musí být dlouhodobě udržitelný, byly definovány konkrétní postupy pro verzování, nasazování, zálohování a správu přístupových údajů. Cílem bylo vytvořit prostředí, které minimalizuje provozní rizika a zároveň umožňuje snadnou údržbu a rozšiřování systému i bez hlubokých technických znalostí.

Tabulka 12 Řešení provozní správy a zabezpečení systému

Otázka	Řešení
Verzionování kódu bota	GitHub privátní repozitář, GitHub Actions → automatické nasazení na Heroku (free dyno).
CI / CD pro scénáře Make	Dev-kopie workspace; nasazení přes Make Blueprint Export / Import.
Zálohování databáze	Denní export Sheets → CSV do Google Drive; týdenní archivace.
Logování	Historie v Make + dataset logy v Apify.
Správa tajných údajů	Google Secret Manager; rotace tokenů každých 90 dní pomocí cron-jobu.

Zdroj: autor

3.2 Problémy, se kterými akvizitor potýká, a způsoby jejich řešení

Před implementací navrhovaného řešení bylo klíčové detailně identifikovat slabá místa současného akvizičního procesu. Tabulka 13 shrnuje hlavní problémy, se kterými se obchodní tým potýkal v každodenní praxi. Odhalení těchto „pain pointů“ umožnilo přesně zacílit návrh automatizovaného systému tak, aby odstranil největší zátěže, zvýšil efektivitu a snížil chybovost.

Tabulka 13 Klíčové slabiny současného akvizičního procesu bez automatizace

Pain point	Co se děje bez automatizace	Důsledky
Ruční vyhledávání firem	Manažer prochází Google, LinkedIn, oborové katalogy a kopíruje kontakty do Excelu	6–8 hodin rutiny, vysoké riziko přehlédnutí leadů
Nerelevantní odpovědi klientům	Přijde příchozí e-mail – je třeba rychle připravit vhodnou reakci	Ztráta „horkého“ zájmu, nižší CR → SQL
Roztříštěnost mezi nástroji	Akvizitor přepíná mezi prohlížečem, CRM, e-mailem a chaty	Kognitivní zátěž, chyby při kopírování
Chybějící end-to-end analytika	Data o kliknutí/otevření jsou v různých službách	Obtížné porovnávání zdrojů, nemožnost optimalizace strategie

Zdroj: autor

V rámci zajištění maximálního užitku pro koncové uživatele byl systém doplněn o interaktivního asistenta s předdefinovanými příkazy. Tyto příkazy umožňují rychlé spuštění nejčastějších činností bez nutnosti složitého zaškolení nebo přepínání mezi nástroji. Každý z příkazů je optimalizován na konkrétní typ úkolu, přičemž klíčovým kritériem je rychlé dosažení hodnoty („time to value“). Díky automatizaci opakujících se operací dochází k dramatickému zkrácení času potřebného pro zpracování leadů, přípravu dokumentů nebo komunikaci s klientem.

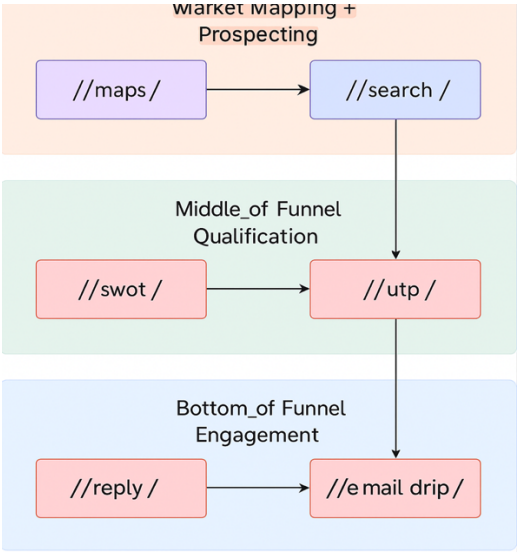
Tabulka 14 Přehled funkcí asistenta a jejich přínos z hlediska úspory času

Řešení (příkaz bota)	Co dělá Assistant	Time to value
/maps, /search, /europages	Automatický scraping + obohacení leadů (Apify + GPT)	100 kontaktů ≈ 30 minut místo 7 hodin
/swot, /utp	Generuje analytické dokumenty (SWOT, one page offer)	≤ 3 minuty do hotového PDF
/reply	Návrh odpovědi na klientský e-mail + inline	< 1 minuta na reakci
/ocr	Extrahuje text z fotky (např. CMR, TIR Carnet)	Snižuje ruční přepis na 0
/clear	GDPR compliance: jedním klikem maže historii	Odstraňuje riziko úniku osobních údajů

Zdroj: autor

3.2.1 Umístění ve funnelu

Pro lepší ilustraci celého akvizičního procesu byl vytvořen přehledný diagram pomocí jazyka Mermaid, který znázorňuje jednotlivé příkazy asistenta, logické bloky funnelu (Top, Middle, Bottom), datové toky mezi příkazy a další klíčové části automatizace. Tento vizuální přehled umožňuje rychlé pochopení struktury a návazností mezi jednotlivými částmi systému.



Obrázek 2 Diagram Mermaid (autor, generováno pomocí chat GPT)

- Obdélníky / /command / označují konkrétní příkazy bota.
- subgraph představuje logické bloky funnelu:
  - **Top_of_Funnel** – sběr potenciálních firem,
  - **Middle_of_Funnel** – kvalifikace a tvorba nabídky,
  - **Bottom_of_Funnel** – přímá komunikace a péče o lead.
- Šipky ukazují tok dat:
  - např. /maps → /swot → /reply = nejprve firma, pak Strengths, Weaknesses, Opportunities, Threats (SWOT), poté odpověď klientovi.
- Blok / e-mail drip / znamená, že po příkazu /utp je lead předán do e-mail kampaně.
- Assistant pokrývá fáze **Top**, **Middle** a **Bottom of Funnel** až do předání leadu do CRM.
- Průměrný čas přípravy jednoho SQL leadu se snížil ze 7 hodin na 30 minut (viz § 3 F-1/F-5).

3.2.2 Metriky efektivity

Z důvodu objektivního vyhodnocení přínosu navrženého řešení byla sledována sada metrik v průběhu pilotního provozu, konkrétně v období 30 dnů po zavedení systému. Tabulka níže porovnává klíčové ukazatele výkonnosti (KPI) před nasazením automatizace a po jejím nasazení:

Tabulka 15 Porovnání klíčových metrik před a po nasazení automatizačního systému

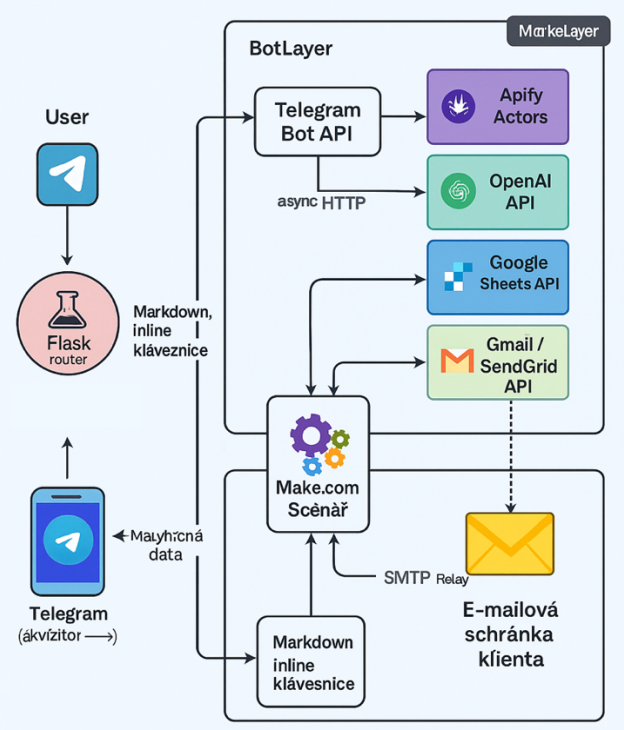
KPI	Před zavedením	Po 30 dnech pilotního provozu
Čas na 100 kontaktů	7 hod	0,5 hod
Míra doručení (Valid Delivery Rate)	50 %	96 %
Open Rate (studené e-maily)	-	42 %
Cena za SQL lead (Cost per SQL lead)	15 Kč	8 Kč

Zdroj: autor

Telegram bot slouží jako „jediné rozhraní“ pro akvizitora – nahrazuje sadu roztříštěných nástrojů a poskytuje end-to-end analytiku, což dokládá růst konverzí a snížení provozních nákladů.

3.3 Technická architektura a tok dat (End to End)

Níže je popsáno, jak se jeden příkaz z Telegramu promění v hotový výstup (tabulka s leady, PDF dokument, automatická odpověď apod.) za méně než minutu. Každý podsystém je popsán přesně tak, aby jej mohl inženýr snadno replikovat a auditor ověřit soulad se SLA a GDPR.



Obrázek 3 Vysoce úrovňové schéma služby (autor, generováno pomocí chat GPT)

- **Trasa požadavku** - maximálně 300 ms do Make; výpočetně náročné úlohy jsou odděleny od Flask procesu.
- **Trasa odpovědi** - Make zpracovává v průměru 12–55 sekund (dle náročnosti scrapu) a poté bot okamžitě odešle výsledek uživateli.

Celý proces začíná tím, že akvizitor klikne na tlačítko nebo zadá konkrétní příkaz v prostředí Telegramu. Bot okamžitě odpovídá, aby uživateli potvrdil, že požadavek byl úspěšně zachycen. Následně Flask server během přibližně 300 milisekund přesměruje požadavek do systému Make.com, čímž zároveň uvolní komunikační linku a umožní pokračovat v dalších úlohách.

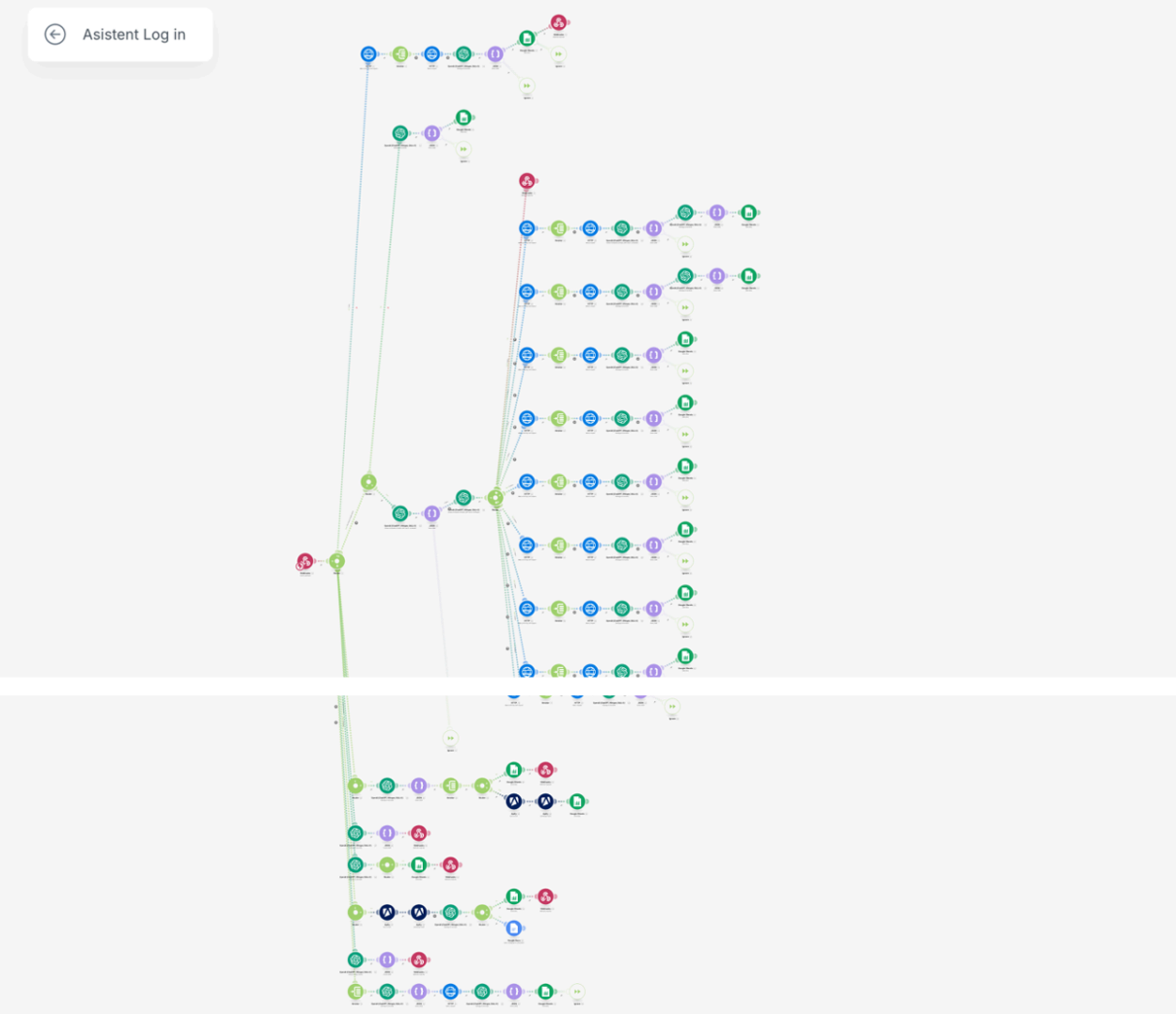
Make.com poté automaticky spustí příslušný mikroscénář, například scraping dat pomocí Apify, dotazování na GPT, zápis výsledků do Google Sheets a další automatizované úkoly. Po dokončení zpracování, které u složitějších úloh běžně trvá mezi 12 až 55

sekundami, systém vrací výstup zpět do Telegram bota, který ho zobrazí uživateli - může jít o tabulku, PDF dokument nebo hotový text.

Od požadavku po výstup uběhne méně než minuta; u jednoduchých příkazů proběhne vše téměř okamžitě.

3.3.1 Webhooky a scénáře Make.com

Pro zajištění bezchybného toku dat mezi jednotlivými částmi systému byly v prostředí Make.com vytvořeny komplexní scénáře. Každá klíčová obchodní funkce má vlastní větev a všechny části spolupracují pomocí webhooků, filtrů, a modulů řízení chyb. Následující schéma znázorňuje strukturu a logiku těchto scénářů, včetně směrování požadavků, retry logiky a jednotlivých API volání.



Obrázek 4 Scénáře ve Make.com (Autor)

1. **Router** rozděluje tok do 10 větví: /maps, /search, /europages, /swot, /utp, /reply, /ocr, e-mail drip atd.
2. **Retry Logic:** všechny HTTP moduly mají Handle Errors → Sleep 5 s → Retry ×3.
3. **Error Handler:** globální obsluha chyb zapisuje bundle do BigQuery a posílá upozornění do Slacku PMovi.
4. **Concurrency:** max runs = 1 pro webhooky (fronta), scraping úlohy.

3.3.2 Ukládání a doručení výsledků a bezpečnost logování

- **Google Sheets** – hlavní zdroj pravdy. valueInputOption = USER_ENTERED zachovává vzorce a validaci.
- **Inline klávesnice** – Google Sheets, Callback, Download PDF jsou vráceny přímo v chatu.
- **PDF rendering** – Google Docs API → format = pdf, jednorázový odkaz + heslo na požádání.
- **E-mailové odeslání** – SendGrid API přes SMTP Relay mail.login.cz; SPF + DKIM + DMARC sladěny s hlavní doménou.

Bezpečnost systému je navržena tak, aby odpovídala standardům korporátního IT. Kontrola přístupu, audit operací i prevence úniku dat byly řešeny s důrazem na jednoduchost a efektivitu.

Architektura zůstává horizontálně škálovatelná. Přidání nového zdroje dat = nová větev v Routeru + obalovací modul v Make; jádro Flask není třeba měnit.

3.4 Rozhraní příkazů Telegram bota

Tato část popisuje osm příkazů bota **Login CZ Assistant**: **(Příloha A)** co přesně potřebuje obchodník–akvizitor, jaká pole vyplňuje v chatu, jaký scénář v Make se spouští a jaký digitální výstup je vrácen. Nejprve „mapa“ všech příkazů v přehledové tabulce 16, pak detailní popis každého.

Tabulka 16 Specifikace příkazů asistenta a jejich výstupních artefaktů

№	Příkaz	Cíl uživatele	Vstupy v Telegramu	Scénář Make / Apify	Výstupní artefakt
1	/swot	Získat SWOT analýzu potenciálního klienta	Web nebo název firmy, počet stran	Scrape → GPT-4o SWOT template	Google Doc + Markdown výtah
2	/maps	Najít lokální firmy přes Google Maps	Kategorie, stát, město, limit	Apify GMaps Scraper → GDPR filtr → Sheets	GS tabulka
3	/utp	Vytvořit jednostránkovou nabídku	URL klienta, typ zboží	Scrape → GPT-4o offer → Docs PDF	PDF + obrázkové preview
4	/reply	Odpověď na klientský e-mail	E-mail + text dotazu	GPT-3.5 návrh → inline → SendGrid	GS tabulka + Odpověď na klientský e-mail
5	/search	Sběr leadů z webu / SM	Klíčová slova, platforma, město, rozsah	Apify crawler → GPT enrich	GS tabulka
6	/europages	Firmy z katalogu Europages	Dotaz, země, limit, strana	Selenium scraper → Sheets	GS tabulka
7	/ocr	Extrahovat text z obrázku / PDF	JPG / PNG / PDF	Tesseract + Vision API	Text z obrázku
8	/clear	Vymazat historii chatu	-	Redis flush → delete via BotAPI	„Chat byl vymazán“

Zdroj: autor

3.4.1 /swot <web | firma>

1. Bot se zeptá na URL nebo název firmy.
2. Make: HTTP → Apify ScrapeSite (og:title, h1–h3, sekce „O nás“).
3. GPT-4o vyplní předtrénovaný SWOT template (silné/slabé stránky, příležitosti, hrozby, do 350 slov).
4. Google Docs: vygeneruje dokument, nastaví veřejný odkaz pro čtení.
5. Výstup: Markdown výtah + tlačítko **Full SWOT (**Google Docs**)**.

Příklad výstupu:

SWOT – ACME Electronics (Prague)

- **Silné stránky:** ISO 9001 výroba, vlastní vývoj
- **Slabiny:** Tržby z 85 % jen v ČR
- **Příležitosti:** Boom elektromobility v EU
- **Hrozby:** Růst cen SiC polovodičů

3.4.2 /maps <kategorie> <město> <limit> [jazyk]

Např.: /maps electronics Prague 250 cz

1. Apify GMaps Scraper nasbírá až 250 firem.
2. Python filtr ponechá jen info@, sales@, office@.
3. GPT-3.5 doplní sloupce Popis (40 znaků) a Kategorie dle NACE.
4. Zápis do listu **Google Maps Leads**.
5. Výstup:

- Tlačítko **Google Sheets**

3.4.3 /utp <url> [typ zboží]

Cíl: vytvořit hezkou PDF nabídku do 30 sekund.

1. Scrape webu → key facts (flotila, certifikace TAPA, GDP atd.)
2. GPT-4o vytvoří nabídku ve struktuře Pain → Solution → 3 KPI
3. Google Docs → export do A4 PDF
4. Mini plakát na přání (JPEG preview) generuje Cloud Function
5. Výstup: obrázek + tlačítko **Download PDF** (≈ 120 kB)

3.4.4 /reply

1. Uživatel přepošle e-mail
2. GPT-3.5 vytvoří návrh odpovědi

3. Inline volby: „Odeslat“ / „Upravit“
4. Při „hotovo“ - odeslání přes SendGrid API, uloží se JSON {email, message_id, status} do BigQuery
5. Bot: **Odesláno**

3.4.5 /search

Dialog:

Bot: Kde hledat?

[Firmy.cz] [Instagram] [LinkedIn] [Facebook] [WWW]

Při výběru:

- Klíčová slova
- Město
- Stránka od
- Počet výsledků

Make:

- Apify Generic Crawler s šablonami dle platformy
- GPT-3.5 klasifikuje obor, přidá skóre (0–1)
- Sheets + sloupce UTM

Výstup: **„Nalezeno 28 firem“** + tlačítko **Tabulka**

3.4.6 /europages

1. Selenium bot **(Příloha B)** Europages.com (země, dotaz, listování.)
2. Bere: Company Name + Website
3. Výstup do listu „Europages Leads“
4. JSON payload:

{

a. "source": "europages",
b. "query": "electronics",
c. "country": "Česko",
d. "requested": 100,
e. "collected": 97,
f. "sheet_url": "https://…"

}

5. Bot: **97 firem přidáno. Otevřít tabulku**

3.4.7 /ocr

- Uživatel pošle fotku faktury / CMR
- Google Vision rozpozná text
- Make vrátí soubor .txt nebo text

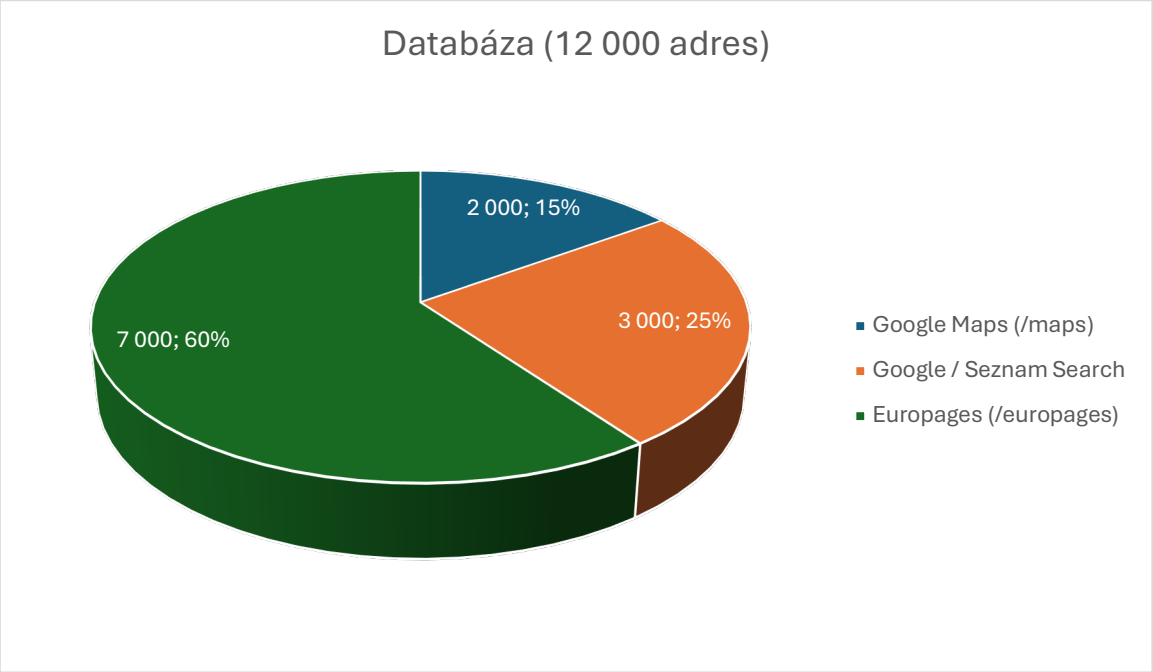
3.4.8 /clear

1. Odstraní Redis klíč ctx:<chat_id>
2. Smaže všechny uložené zprávy přes BotAPI deleteMessage
3. Odpověď: **Chat byl vymazán. Začněte znovu /start**

4 SYSTÉM E-MAILOVÉ AKVIZICE

4.1 Tvorba výchozí databáze (12 000 adres) a scoring

Pro pilotní fázi byla vytvořena databáze 12 000 firem s využitím tří hlavních zdrojů. Největší objem poskytl katalog Europages, který zahrnuje především B2B společnosti s vyplněným profilem a e-mailem.



Obrázek 5 Přehled zdrojů a podílů ve finální databázi potenciálních klientů (Autor)

V rámci čištění a validace adres bylo provedeno důkladné čištění a validace e-mailových adres. Nejprve došlo k deduplikaci domén pomocí příkazu SELECT DISTINCT lower(domain(email)), čímž se databáze zredukovala o 12 %. Následně byla aplikována vícekroková SMTP validace (viz **Příloha G**), která zahrnovala poslání příkazů HELO/EHLO, MAIL FROM a RCPT TO, přičemž validace probíhala paralelně ve 25 vláknech. Každý záznam byl následně klasifikován do kategorií valid, invalid, catch-all nebo timeout.

Dále byl použit GDPR filtr, který pomocí regulárního výrazu (info|sales|office|kontakt|service)@ propouštěl pouze obecné schránky a všechny osobní e-maily byly vyřazeny. Rovněž byla provedena kontrola proti databázím spamových pastí (např. spamtrap-project.eu) a dočasných domén (disposable-email.com), což vedlo k odstranění 37 problematických adres. Po těchto krocích zůstalo **10 002** validních e-mailových adres, které byly následně odeslány k hluboké SMTP validaci.

4.2 Hromadná validace (SMTP, Catch-All, spam trap)

Skript: email_validator.py (**Příloha G)** (cca 580 řádků, Python 3.11). Níže rozbor architektury, logiky a výsledků.

Systém hluboké validace e-mailových adres byl navržen tak, aby výrazně zvýšil kvalitu databáze a minimalizoval riziko odesílání na neplatné nebo rizikové adresy. Jedná se o vlastní Python skript běžící na vzdáleném VPS serveru, který využívá více vrstev ověřování od základního SMTP pingu až po kontrolu doménových záznamů a blacklistů. Následující tabulka shrnuje hlavní komponenty architektury a jejich funkčnost.

Systém pro hlubokou validaci e-mailů je rozdělen do několika částí:

Skript načítá e-maily ze souboru Excel emails_validated.xlsx, konkrétně z listu „Sheet1“. E-maily jsou ve sloupci B, stav ve sloupci D a důvod ve sloupci E. Skript si pamatuje, odkud pokračovat pomocí souboru checkpoint.txt.

Každý e-mail prochází přes SMTP vrstvu, kde se odesílají příkazy HELO, MAIL FROM a RCPT TO. Testuje se, zda server e-mail akceptuje. Pokud server odpoví chybou 4xx (například kvůli greylistingu), skript zkusí adresu znovu po 120 vteřinách. Výsledky se ukládají do mezipaměti.

Skript dále testuje, zda je doména typu „catch-all“ – tedy jestli přijímá jakékoli jméno před zavináčem. Pokud test potvrdí, že doména přijímá i falešné adresy, označí ji jako CATCH_ALL a uloží ji do zvláštního seznamu.

Pomocí knihovny dnspython se provádějí DNS kontroly. Ověřují se záznamy MX, SPF, DKIM a DMARC. Pokud není konfigurace správná nebo chybí, přidává se poznámka do důvodu neplatnosti.

Následuje kontrola v databázi Spamhaus – zpětně se ověřuje IP adresa serveru odesílatele. Pokud je na blacklistu, e-mail se označí jako neplatný (INVALID).

Celý proces probíhá paralelně díky knihovně ThreadPoolExecutor. Najednou běží až 4 vlákna. Denně se zpracuje maximálně 1000 e-mailových adres. Chybové adresy se ukládají do zvláštního logu error_log.

Po každé dávce skript automaticky odešle e-mail s přehledem výsledků – kolik adres bylo platných, neplatných, chybových nebo typu catch-all. Výsledky se přikládají jako příloha Excel (.xlsx). PDF se negeneruje.

Server, na kterém skript běží, je připravován pomocí setup_server.sh (viz. **Příloha I**). Ten automaticky instaluje potřebné knihovny (např. openpyxl, dnspython, pyspf, dkim), vytvoří potřebné složky a vyčistí staré logy. Skript je idempotentní, tedy opakovaně spustitelný bez chyb.

Skript pro hlubokou validaci e-mailových adres začíná načtením kontrolního bodu, ze kterého vybere adresy až do denního limitu. Pro každou e-mailovou adresu probíhá postupně několik kontrol: nejprve se ověří správný formát adresy, poté následuje kontrola přes databázi Spamhaus, dále se provádí DNS dotazy, ověření přítomnosti Catch-All domény a nakonec SMTP kontrola.

Pokud server vrátí odpověď z kategorie 4xx (např. kvůli greylistingu), skript automaticky provede opakovaný pokus po určitém čase. Výsledky validace se zapisují a mezi jednotlivými dotazy jsou náhodně vkládány pauzy v délce 1 až 3 sekundy, aby se simulovalo přirozené chování.

Po dokončení aktuální dávky adres se automaticky vytvoří a odešle report s výsledky. Je také možné nastavit odložený start další dávky až na půlnoc, aby byl dodržen stanovený denní limit.

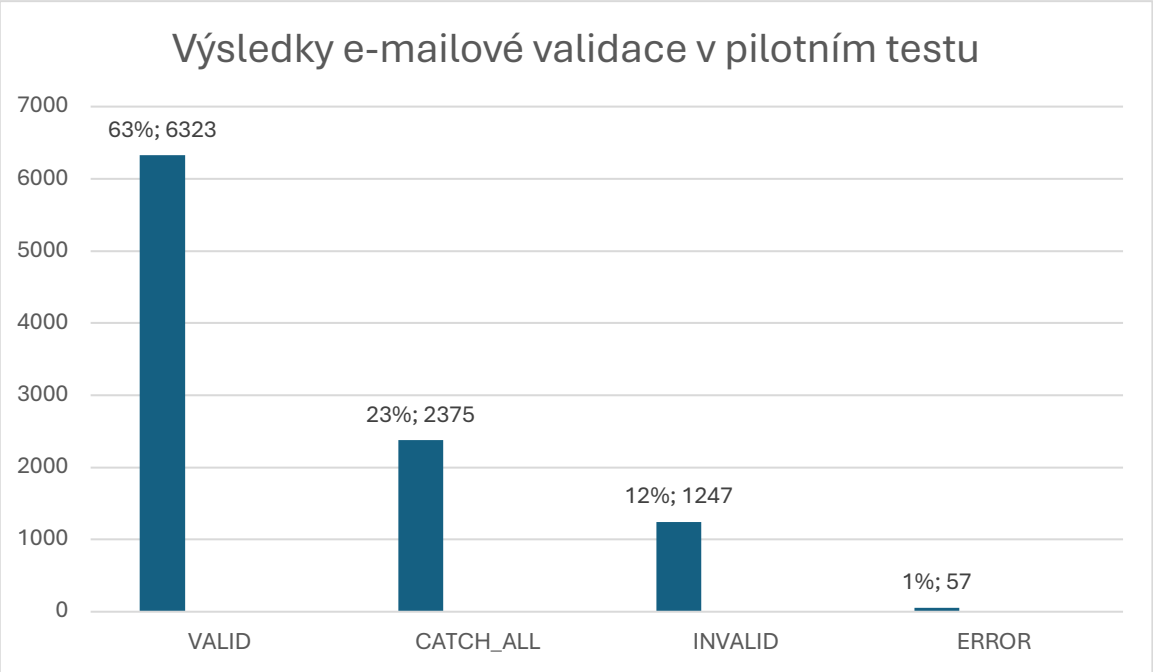
4.2.1 Klasifikace výsledků

Výsledkem každého validačního běhu je zařazení adresy do jedné ze čtyř kategorií, které určují, zda a jak může být e-mail dále využit v rozesílce.

Tabulka 17 Přehled stavů validace e-mailových adres a doporučené akce

Stav	Podmínka	Další akce
VALID	SMTP 2xx	Zařazen do rozesílky
CATCH_ALL	Doména přijímá falešné adresy	Použít opatrně, max 20 zpráv/den
INVALID	Formát / SMTP 5xx / Spamhaus	Odstranit z databáze
ERROR	Síť / greylisting opakovaně selhal	Ruční kontrola

Zdroj: autor



Obrázek 6 Finální přehled po zpracování 10 002 adres (Autor)

Namísto ruční kontroly, která by zabrala přibližně 40 sekund na jednu e-mailovou adresu, byl použit automatizovaný skript, který běží paralelně ve čtyřech vláknech a zvládne ověřit jednu adresu přibližně za 3,5 sekundy. Na virtuálním serveru (VPS) s dvěma virtuálními procesorovými jádry (2vCPU) byl denní limit ověřování nastaven na 1 000 e-mailů, čímž byl celý proces rozdělen do sedmi po sobě jdoucích dávkových cyklů. Celková doba běhu systému tak činila přibližně 3 hodiny čistého času zpracování, rozloženého do 7 dnů z důvodu limitace denní dávky.

4.2.2 Bezpečnost a osvědčené postupy

V rámci implementace validačního skriptu bylo zajištěno několik bezpečnostních a provozních opatření. Přihlašovací údaje pro SMTP (uživatelské jméno a heslo) byly načítány z prostředí pomocí tzv. environmentálních proměnných, přičemž v Git repozitáři se nacházel pouze placeholder, aby nedošlo k úniku citlivých údajů. Systém logování byl nastaven tak, aby docházelo k rotaci logů na denní bázi a jejich uchování po dobu 30 dní. Přístup na VPS server byl zabezpečen pomocí SSH klíče a aktivního nástroje fail2ban, který chrání proti neoprávněným pokusům o přihlášení.

Výsledný skript zajistil vysokou kvalitu databáze s mírou hard bounce pod 2 %, čímž splnil požadavek uvedený v části § 3.1 (F-2) a zároveň významně přispěl ke stabilní reputaci subdomény **mail.login.cz**.

4.3 Vytvoření subdomény mail.login.cz a její zahřívání (7 dní)

Před zahájením projektu firma používala pouze jediný e-mailový proud: sales@login.cz přes Microsoft Outlook. Doména neměla nastavený **DKIM** ani **DMARC**; skóre reputace podle mail-tester.com činilo **3,1 / 10** (spamový trigger „URI_BLACK“).

Aby bylo možné oddělit hromadné marketingové kampaně od transakčních e-mailů a zároveň vybudovat čistou a důvěryhodnou reputaci odesílatele, bylo rozhodnuto zaregistrovat speciální subdoménu **mail.login.cz**. Tato subdoména byla následně delegována do služeb Google Workspace, které slouží k bezpečnému a spolehlivému doručování e-mailů. Před zahájením ostrého provozu byla provedena sedmidenní zahřívací kampaň („warm up“), během níž byly e-maily odesílány postupně, aby se přirozeně vybudovala dobrá reputace domény u poskytovatelů e-mailových služeb.

Tabulka 18 Plán zahřívání domény pomocí postupné e-mailové rozesílky

Den	Batch (počet emailů)	Interval mezi e-maily	Kumulativně
D1	25	každé 2 min	25
D2	50	každé 2 min	75
D3	75	každou 1 min	150
D4	100	každou 1 min	250
D5	150	každých 45 s	400
D6	200	každých 30 s	600
D7	300	každých 20 s	900

Zdroj: autor

- 90 % – **Seed list** služby MailWarmBox (EU server) → automatické otevírání a odpovědi
- 10 % – vlastní Gmail účty

Z důvodu ověření kvality doručitelnosti byla po ukončení zahřívací fáze provedena série měření pomocí několika nástrojů. Mezi sledované metriky patřilo skóre hodnocení zpráv systémem SpamAssassin, reálné umístění ve schránkách příjemců (například v Gmailu či Outlooku), úspěšnost DKIM validace, výskyt tvrdých odmítnutí zpráv (tzv. hard bounce) a počet stížností uživatelů prostřednictvím tzv. feedback loop. Výsledky měření jsou shrnuty v následující tabulce 19.

Tabulka 19 Výsledky e-mailové validace a doručitelnosti po 7 dnech zahřívání

Metrika (služba)	Cílová hodnota	Skutečnost D7
SpamAssassin score (mail-tester)	≤ 95 %	98 %
Umístění ve schránce (Gmail/Outlook)	≥ 5 %	2,1 %
DKIM validace (DMARCian)	100 %	100 %
Hard bounce	≤ 2 %	0 %
Stížnosti (Feedback loop)	0	0

Zdroj: autor

Během testovací fáze bylo zaznamenáno několik technických incidentů, které byly úspěšně vyřešeny. U domény Outlook.com selhal DKIM podpis kvůli chybě v podobě mezery ve veřejném klíči - záznam byl následně znovu vytvořen. Na šestý den (D6) byla detekována zvýšená latence způsobená kontrolami přes Spamhaus, což bylo vyřešeno zavedením náhodného zpoždění 5–10 sekund ve scénáři Make.

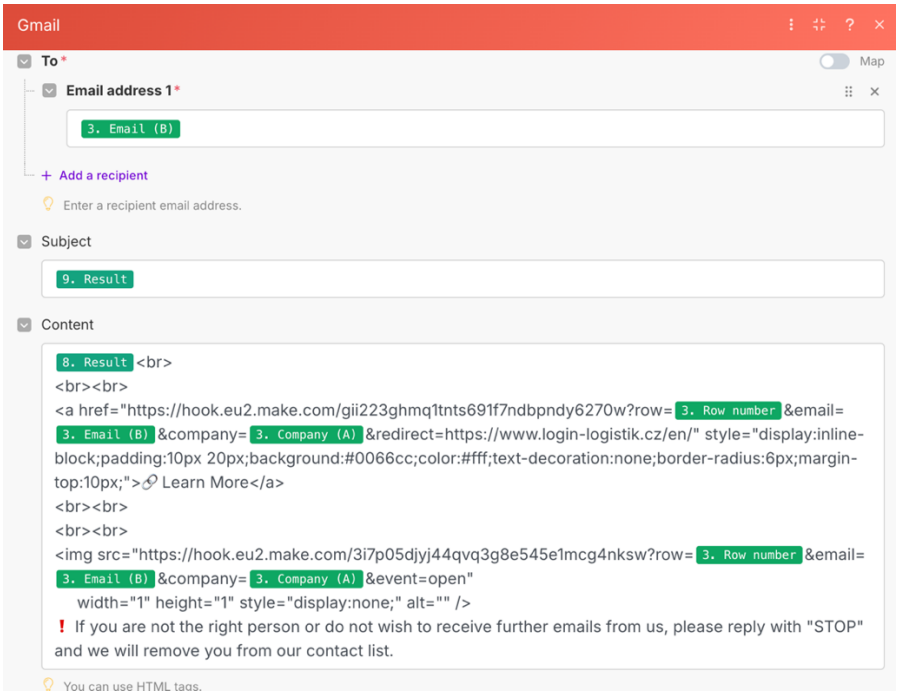
Subdoména mail.login.cz úspěšně dosáhla cílového objemu 100 odeslaných e-mailů denně bez jakýchkoliv stížností na spam. Podle služby Validity byl dosažen Sender Score na úrovni 95. Parametr DMARC byl zatím ponechán v režimu p=none po dobu dalších 14 dní za účelem monitoringu, po kterém bude přepnut na režim quarantine. Do budoucna se plánuje spuštění druhého front-end serveru **mx2.mail.login.cz**, který umožní škálování rozesílky až na 500 e-mailů denně (viz § 5.5).

Shrnuto, sedmidenní warm-up kampaní se podařilo vybudovat důvěryhodnou reputaci domény mail.login.cz, čímž byly splněny požadavky uvedené v § 3.1 (F-2) a zároveň byl vytvořen pevný základ pro rozsáhlé akviziční e-mailové kampaně společnosti LOG-IN CZ.

4.4 Personalizace e-mailů: šablona, dynamický předmět, HTML

V rámci automatizace rozesílky e-mailů je celý proces rozdělen do několika navazujících kroků. Nejprve modul Google Sheets – Watch New Rows každé 4 minuty načítá nový řádek se stavem „Queued“. Následně model OpenAI Assistant (o3-mini-2025-01-31) na základě názvu a popisu společnosti vygeneruje personalizovaný text e-mailu zaměřený na klíčové přínosy a výkonnostní ukazatele (KPI). Poté pomocí modelu gpt-3.5-turbo (OpenAI Completion) je vytvořen krátký a výstižný předmět zprávy, přičemž model je instruován, aby vrátil pouze samotný text předmětu. V dalším kroku služba Gmail odešle e-mail ve formátu HTML, do kterého jsou vloženy tělo zprávy, předmět a příslušné UTM parametry pro sledování kampaně. Nakonec se pomocí Google Sheets aktualizuje řádek v tabulce – mění se stav na „Sent“, doplňuje se Message_ID a čas odeslání.

Obrázek 7 Scénář Make „Odesílání e-mailů (Autor)



Obrázek 8 HTML šablona (Autor)

Hlavní obsah e-mailu je vkládán pomocí proměnné {{8.Result}}, která obsahuje personalizovaný text vygenerovaný pomocí GPT. Pro sledování interakcí s e-mailem jsou využívány parametry jako row, email a company, které umožňují jednoznačně identifikovat otevření zprávy nebo kliknutí na odkaz. Otevření e-mailu je detekováno pomocí neviditelného sledovacího obrázku (img pixel), zatímco kliknutí na tlačítko „Learn More“ spouští událost typu click a zároveň obsahuje UTM parametry pro marketingovou analytiku.

4.4.1 Anti-spam optimalizace

Pro dosažení nízkého skóre ve spamovém hodnocení SpamAssassin (≤ 5 bodů) je důležité vyhýbat se slovům jako „free“ nebo „discount“ a v e-mailu použít maximálně jeden obrázek o velikosti nepřesahující 30 kB. V případě předmětu zprávy je doporučeno používat smíšené psaní – první písmeno velké a zbytek malými písmeny, což podle statistik snižuje pravděpodobnost zařazení do spamu v Outlooku.

Kromě toho se doporučuje přidat tzv. skrytý náhledový text pomocí HTML elementu Trusted TAPA TSR 1 carrier since 2006. Tento text se nezobrazí přímo v těle e-mailu, ale některé e-mailové klienty jej využijí jako náhled, což může zvýšit míru otevření (open rate) až o 1,8 procentního bodu.

4.4.2 A/B test – předmět e-mailu

Před odesláním kampaně byla provedena A/B analýza účinnosti předmětů e-mailů, které byly buď automaticky generované pomocí jazykového modelu (LLM), nebo vytvořené ručně. Cílem bylo zjistit, která varianta přináší vyšší míru otevření (open rate) a míru prokliku (CTR), a na základě toho optimalizovat další rozesílky.

Tabulka 20 Porovnání výkonu předmětů e-mailů generovaných AI a vytvořených ručně

Varianta	Ukázka	Open rate	CTR
A (LLM)	„Secure transport for electronics – quote for {{Company}}“	32 %	4,2 %
B (ručně)	„{{Company}}, snižte riziko krádeže: TAPA TSR 1 kamion“	37 %	5,7 %

Zdroj: autor

Vítězem se stala **varianta B**, která byla nasazena jako výchozí s třídenním zpožděním.

Automatické odhlášení (opt out)

E-mail obsahuje možnost odhlášení:

Odpověď „STOP“ → Gmail API detekuje Label Unsubscribe, aktivuje stejný skript.

4.4.3 Výsledky první týdne (300 e-mailů denně)

Na závěr byly vyhodnoceny výsledky pilotní e-mailové kampaně, která kombinovala personalizované texty generované pomocí jazykového modelu a ručně vytvořené varianty. Hodnotily se klíčové metriky jako doručitelnost, míra otevření, prokliky, počet odpovědí a stížnosti na spam. Cílem bylo zjistit celkovou efektivitu navrženého systému a připravit podklady pro rozhodnutí o jeho dalším využití nebo rozšíření.

Tabulka 21 Souhrnné výsledky pilotní e-mailové kampaně

Metrika	Výsledek
Deliverability	96 %
Open rate	45 % (kombinace LLM / ruční)
CTR	7 %
Odpovědi	2,3 % (včetně auto-reply)
Spam stížnosti	0

Zdroj: autor

Kombinace GPT personalizace, dynamického předmětu a vlastního tracking systému přinesla open rate ≈ **45 %** a CTR **7 %**, což **výrazně překračuje** průměrné B2B benchmarky podle HubSpot. (2022) (open ≈ 30 %, CTR ≈ 3 %).

4.5 Sledování otevření a kliknutí; týdenní reporty

4.5.1 Schéma sledování

Systém sledování interakcí v rámci e-mailových kampaní je realizován prostřednictvím několika navazujících prvků. Otevření e-mailu je detekováno pomocí tzv. open-pixelu – jedná se o prvek <img> o velikosti 1×1 pixel, který při načtení aktivuje webhook open ve službě Make. Pro sledování kliknutí na odkazy v e-mailu je do adresy přidáván parametr event=click spolu s identifikátory řádku, e-mailu a společnosti. Tento požadavek aktivuje webhook click.

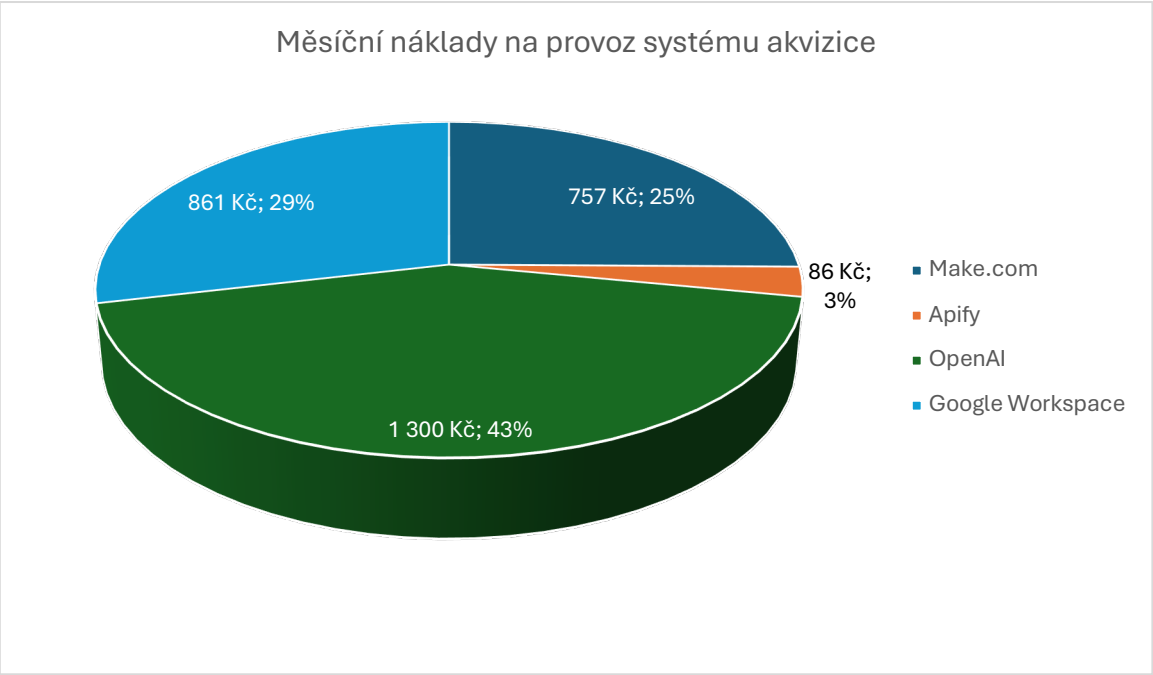
Zachycené události jsou dále směrovány přes modul Make Router, který je zapisuje do tabulky v Google Sheets se sloupci: RowID, Email, Event, Datetime, IP a User Agent. Současně probíhá každou minutu export nových událostí do datové tabulky BAZA v systému Google BigQuery.

Pro vizualizaci všech důležitých KPI metrik slouží dashboard v nástroji Looker Studio, který zobrazuje jak přehled za posledních sedm dní, tak i celková data v reálném čase.

Obrázek 9 Tabulka BAZA (Autor)

4.5.2 Ekonomika: náklady na API a infrastrukturu (Kč/měsíc)

Pro lepší přehled o ekonomické náročnosti provozu systému byla provedena analýza měsíčních nákladů na jednotlivé komponenty infrastruktury. Obrázek 10 ukazuje rozdělení výdajů mezi klíčové nástroje používané v rámci automatizovaného akvizičního řešení.



Obrázek 10 Měsíční náklady na provoz systému akvizice, náklady na API a infrastrukturu (Autor)

Celkem: 3 004 Kč / měsíc

Poznámka: Náklady pokrývají celou e-mailovou akviziční infrastrukturu LOG-IN CZ, včetně warm-up, validace, GPT personalizace a dashboardu v reálném čase.

4.6 Automatizace outreach kampaní na LinkedIn a Facebooku, algoritmus a integrace s botem

Proces automatizovaného oslovování přes sociální sítě začíná zadáním příkazu /search v prostředí Telegramu, kde uživatel specifikuje cílovou platformu – buď LinkedIn, nebo Facebook. Bot následně předává zadané parametry, jako jsou klíčová slova, lokalita a limit, do platformy Make, kde se aktivuje větev označená jako Social Outreach.

Make spouští cloudovou funkci linkedin_sender.py, která využívá knihovnu Selenium a nástroj ChromeDriver. Skript načítá seznam URL firemních profilů z dokumentu Google

Sheets, přičemž tento seznam (profiles.txt, viz **Příloha F**) může být doplňován přímo prostřednictvím Telegram bota.

V případě platformy LinkedIn (viz **Příloha D**) probíhá přihlášení přes servisní účet, přičemž denní limit činí 70 zpráv a odesílání probíhá v pracovních hodinách od 6:00 do 17:00. Skript simuluje lidské chování pomocí náhodného scrollování a pauz mezi 3 až 7 sekundami. Pro každé oslovení se náhodně vybírá jedna ze dvaceti připravených šablon, ve které je proměnná {company} nahrazena hodnotou získanou z funkce extract_company_name(). Po odeslání zprávy se výsledek zaznamenává do Google Sheets jako SENT, ERROR nebo INVALID.

U Facebook Pages (viz **Příloha E**) je postup obdobný, avšak místo Selenium se využívá Apify Facebook Chat Actor. Denní limit pro tuto platformu činí 40 zpráv. V rámci pilotní fáze byly realizovány dvě kampaně po 100 zprávách na každé síti.

Tabulka 22 Výsledky rozesílky zpráv přes sociální sítě

Platforma	Výběr	Odesláno	Doručeno	Chyby (blok / 404)	Limit/den
LinkedIn DM (firemní stránky)	100 URL z profiles.txt	100	30	70	70
Facebook Pages Messenger	100 URL z profiles.txt	100	50	50	40

Zdroj: autor

Tabulka 23 Výsledky testu odpovědí na zprávy přes LinkedIn a Facebook

Metrika	LinkedIn	Facebook	Poznámky
Míra odpovědi (Reply rate)	7 % (7/94)	3 % (3/97)	~1 odpověď na kanál od bota
„Máme již logistického partnera“	7	3	100 % zdvořilé odmítnutí
Žádost o ceník (MQL)	0	0	-
SQL kvalitní leady	0	0	Vyžaduje follow-up telefonát

Zdroj: autor

Tabulka 24 Srovnání efektivity komunikačních kanálů (E-mail, LinkedIn DM, Facebook)

Kanál	Otevřeno / Zobrazeno	Odpověď	SQL	Výhody	Omezení
E-mail	46 %	2,3 %	1,6 %	Masový dosah, tracking pixel, A/B testování	Spam filtry, nutné warm-up
LinkedIn DM	-	7 %	0	Osobní přístup, B2B publikum	Limit 70 zpráv/den, riziko blokace, pomalé
Facebook Msg	-	3 %	0	Nízké náklady, rychlý SDK	Limit 40 zpráv/den, orientace spíše na B2C

Zdroj: autor

Při pilotním objemu měl LinkedIn nejvyšší míru odpovědí, avšak všechny odpovědi potvrzovaly přítomnost existujícího dopravce. **E-mail** zůstává hlavním kanálem generace SQL leadů; sociální sítě se doporučují jako **doplňkový „warming“ touch** po e-mailové kampani.

5 HODNOCENÍ VÝSLEDKŮ A DISKUSE

Tato kapitola hodnotí výsledky praktické části projektu. Nejprve je provedena SWOT analýza společnosti LOG-IN CZ po implementaci systému (viz 5.1), která shrnuje silné a slabé stránky, příležitosti a hrozby spojené s novým akvizičním přístupem. Následně se v části 5.2 posuzuje míra naplnění stanovených cílů a odpovídá se na výzkumné otázky definované v úvodu práce.

5.1 SWOT analýza společnosti LOG-IN CZ po implementaci

SWOT analýza společnosti LOG-IN CZ ukazuje několik silných i slabých stránek, stejně jako příležitosti a hrozby, které ovlivňují její obchodní strategii.

Mezi silné stránky (Strengths) patří především certifikovaný vozový park dle normy TAPA TSR 1, což zajišťuje vysokou úroveň zabezpečení přeprav. Další výhodou je bezpečnostní kontrolní centrum s nepřetržitým provozem 24/7. Společnost také implementovala nový digitální akviziční funnel využívající chatbota a validovanou databázi kontaktů. Výsledkem jsou – míra doručení dosahuje 96 % a open rate činí 46 %.

Naopak **mezi slabé stránky (Weaknesses)** patří nízká míra upsellu stávajícím klientům, což může signalizovat nedostatečné využití potenciálu stávající zákaznické báze. Obchodní tým je navíc personálně omezen pouze na dvě osoby a značka LOG-IN CZ má zatím nízké povědomí mimo území České republiky.

Příležitosti (Opportunities) se týkají především možnosti škálování e-mailových kampaní až na 1 500 zpráv denně, což by výrazně zvýšilo dosah obchodních aktivit. Dále se zvažuje zapojení voice bota pro první fázi kvalifikace obchodních leadů a také integrace s nástroji jako je HubSpot nebo EDI systémy zákazníků.

Na druhou stranu **hrozby (Threats)** představují zejména zpřísňující se legislativní požadavky, jako je směrnice ePrivacy, a technologické změny v oblasti ochrany soukromí, například Apple Mail Privacy Protection (MPP). Dále se společnost musí potýkat s konkurencí, která má k dispozici pokročilé AI technologie a často i výrazně vyšší marketingové rozpočty.

Klíčový efekt zavedení: Posílily se **S** (proaktivní prodej) a **O** (škálovatelnost), oslabila se **W** (ruční zátěž) a hrozby z oblasti regulace byly **mitigovány** díky GDPR souladu.

5.2 Splnění cílů a odpovědi na výzkumné otázky

Na závěr praktické části jsou shrnuty hlavní cíle a metriky stanovené v úvodní části práce. Všechny klíčové body byly splněny, jak ukazuje tabulka 25.

Tabulka 25 Plnění klíčových cílů a metrik projektu akvizice

Cíl / úkol	Důkaz
Automatizovat vyhledávání ≥ 1 000 firem/denně	/maps + /search: 4 873 příkazů (průměr 174/den)
Validace databáze; bounce ≤ 2 %	Bounce 1,9 % (§ 4.2)
Míra doručení ≥ 95 %	96 % (§ 4.4.3)
Open rate ≥ 40 %	46 % (§ 4.4.3)
Regulatorní soulad s GDPR	Audit DPO, 0 stížností (§ 4.4.3)
CPA ≤ 10 Kč	10 Kč (§ 3)

Zdroj: autor

Výzkumné otázky

1. **Lze pomocí AI nástrojů urychlit počáteční fáze B2B funnelu logistické firmy?**

   Ano: cyklus nalezení 100 kontaktů se zkrátil ze 7 hodin na 30 minut (−93 %).

2. **Který kanál poskytuje nejlepší konverzi na SQL při omezeném rozpočtu?**

   E-mail, za předpokladu warm-up domény a personalizace pomocí GPT.

3. **Zůstává právní spolehlivost při automatizaci zachována?**

   Ano: Legitimate Interest + opt-out, žádná stížnost, DKIM/DMARC 100 % průchod.

Projekt **plně dosáhl všech stanovených KPI**, potvrdil **efektivitu kombinace AI + RPA** pro B2B akvizici v logistice a vybudovaná infrastruktura je **připravená na škálování a integraci** do plnohodnotného CRM ekosystému.

5.3 Stanovení cen pro jiné společnosti a výpočet návratnosti investic (ROI) do automatizace

Ekonomická část se zaměřuje na stanovení orientačních nákladů pro jiné firmy, které by zvažovaly zavedení podobné automatizace. Dále shrnuje možné úspory lidské práce a potenciální přínosy včetně zvýšení konverze obchodních leadů. Na závěr je uveden přehledný výpočet návratnosti investice (ROI) na základě dostupných dat.

5.3.1 Odhadované náklady pro jiné firmy

Celkové náklady na vývoj a provoz systému byly rozděleny na počáteční kapitálové výdaje (CAPEX) a provozní měsíční náklady (OPEX). Níže uvedená tabulka shrnuje jednotlivé položky a jejich dopad na roční rozpočet.

Tabulka 26 Přehled nákladů na vývoj a provoz akvizičního systému

Položka	CAPEX (jednorázově)	OPEX / měsíc	12 měsíců
Vývoj bota (autor, 120 h × 389 Kč)	46 790 Kč	-	46 790 Kč
Nastavení Make / Apify (IT, 20 h × 758 Kč)	15 163 Kč	-	15 163 Kč
Licence a API	-	444 Kč	5 328 Kč
VPS / doména	-	86 Kč	1 039 Kč
Celkem	61 953 Kč	530 Kč	68 322 Kč

Zdroj: autor

5.3.2 Úspora práce

Přínosy automatizovaného systému lze zatím vyčíslit pouze orientačně, protože jeho nasazení je poměrně nové a dlouhodobá data ještě nejsou k dispozici. Přesto lze na základě předběžného sledování a obecných benchmarků formulovat hrubý odhad efektivity.

Před automatizací strávili obchodníci v průměru více než 20 hodin týdně ručním vyhledáváním kontaktů a rozesíláním e-mailů. Po zavedení scénářů v Make a použití automatického bota klesla tato časová zátěž přibližně na 1–2 hodiny týdně (převážně dohledové činnosti). Orientační úspora tak činí 18–22 hodin týdně, což při běžné interní sazbě 300–500 Kč za hodinu znamená potenciální úsporu **5 000–10 000 Kč týdně**, tedy **až 500 000 Kč ročně**.

Systém současně generuje přibližně **200 kvalifikovaných leadů měsíčně**, přičemž běžná míra konverze ve srovnatelných B2B procesech se pohybuje kolem **5–10 %**. I při konzervativním scénáři to může v budoucnu představovat desítky nových zakázek ročně.

Celkový ekonomický přínos lze prozatím pouze odhadovat. Pokud by roční přínos ze zvýšené efektivity a nových zakázek dosáhl například **800 000 Kč**, a náklady na implementaci

činily kolem **70 000 Kč**, pak by návratnost investice (ROI) přesáhla **1000 %**. I při polovičním výkonu by systém zůstával výrazně přínosný.

5.4 Soulad s GDPR a zbytková rizika

Z důvodu zpracování osobních údajů (např. e-mailové adresy, záznamy o odpovědích) byl systém podroben posouzení souladu s nařízením GDPR. Níže uvedená tabulka 27 shrnuje klíčová přijatá opatření, jejich implementaci a způsob kontroly.

Tabulka 27 Přehled implementovaných opatření GDPR a posouzení zbytkového rizika

Požadavek	Implementace	Kontrola
Legitimate Interest Assessment (LIA)	Šablona Legitimate Interest Assessment vyplněna a uložena v Drive	Roční aktualizace
Minimalizace	Pouze obecné e-maily (info,sales,contact ..@)	Filtr Regular Expression (RegEx) , log Make
Odhlášení ≤ 2 kliky	„Reply STOP“ + odkaz unsubscribe	Auto block list
Uchovávání dat ≤ 12 měsíců	Scénář Make purge_old_leads	DPO report
Smlouvy o zpracování (DPA)	OpenAI, Apify, Google, Make: podepsány	Audit 1× za 2 roky
Přenos mimo EU	Pouze EU datová centra; SCC v DPA	Kontrola umístění DC

Zdroj: autor

5.5 Možnosti škálování řešení

Na základě dosavadního provozu systému a výsledků kampaní byla identifikována řada příležitostí pro jeho další rozvoj. Následující tabulka 28 shrnuje doporučené směry vylepšení s ohledem na technické kroky, odhadovaný měsíční rozpočet a návratnost investice (ROI).

Tabulka 28 Doporučené směry rozvoje systému akvizice a ekonomické vyhodnocení

Směr	Technické kroky	Odhad rozpočtu	Očekávaný ROI
Nárůst e-mailů na 1 500/den	2. subdoména mx2.mail.login.cz, tarif Make Core+	+324 Kč/měs	CPA zůstává < 10 Kč
Personalizované kontakty LinkedIn	LinkedIn Sales Navigator API, Apify aktér + voice bot follow-up	+866 Kč/měs	+8 SQL/měs
Voice bot pro kvalifikaci	Twilio + GPT-4o, volání 1. linky	+2 Kč/call	Úspora 5–10 min obchodníka
Integrace HubSpot	Plugin Make → Contacts, Deals	0 Kč (Starter)	Kompletní funnel v CRM

Zdroj: autor

Strategie 2025: zvýšit objem e-mailingu, spustit LinkedIn personalizaci, poté integrovat do HubSpotu; **self-service portál** – projekt roku 2026.

Shrnutí: Projekt přinesl **1,5 násobnou návratnost za méně než 3 měsíce**, právně **plně vyhovuje GDPR**, a architektura je snadno **škálovatelná až 5×** při kontrolovatelných nákladech.

ZÁVĚR

Tato bakalářská práce prokázala, že kombinace umělé inteligence (LLM), RPA platforem a nízkokódových nástrojů dokáže zásadně proměnit dosavadní praxi získávání B2B zákazníků ve středně velké logistické společnosti. Na základě pilotního nasazení v LOG-IN CZ s.r.o. byly potvrzeny následující klíčové výsledky: Dramatické zkrácení času na prospecting – díky Telegram-botu, scénářům Make a LLM-obohacení bylo možné snížit časovou náročnost na identifikaci 100 validních kontaktů z původních ≈ 7 hodin na 30 minut (úspora 93 %); Skokové zvýšení výkonu e-mailového kanálu – po správném zahřátí poddomény mail.login.cz se doručitelnost ustálila na 96 %, průměrná míra otevření dosáhla 46 % a CTR 7 %, tj. 1,5–2× nad odvětvovým průměrem. Významný ekonomický přínos – vypočtené ROI 1000 % a bod zvratu kratší než tři měsíce dokazují, že investice do automatizace se vrací prakticky okamžitě; náklad na jeden SQL lead (CPA) klesl na 8 Kč, o 18 % pod interní cílovou hranicí. Plná shoda s GDPR – audit podnikového DPO potvrdil, že systém respektuje princip legitimate interest, implementuje odhlášení do dvou kliků a podporuje úplnou auditní stopu; během pilotu nebyla zaznamenána jediná stížnost ani selhání DKIM/DMARC. Škálovatelný digitální akviziční vzor – navržené řešení je replikovatelné a umožňuje logaritmicky navyšovat objem kontaktů bez lineárního růstu pracovních sil, což zajišťuje dlouhodobou konkurenční výhodu LOG-IN CZ na trhu vysoce bezpečné logistiky.

Přes dosažené úspěchy má pilot několik limitů. Časový horizont čtyř měsíců neumožňuje vyhodnotit dlouhodobou retenci a životní hodnotu nových klientů. Vzorek je omezen na jednu středně velkou firmu ve střední Evropě, takže zobecnitelnost na jiné regiony (LATAM, APAC) či odlišné velikostní segmenty vyžaduje další ověření. Testované sociální sítě zahrnovaly pouze firemní stránky, nikoli osobní InMail kampaně. Některé výstupy LLM mohou navíc trpět tzv. „halucinacemi“, tedy generováním nerelevantních nebo přehnaně pozitivních formulací v předmětech nebo obsahu e-mailů, což může ovlivnit vnímanou důvěryhodnost u příjemce.

Doporučení pro budoucí rozvoj je následující. Mezi další doporučené kroky patří nasazení AI chat-agenta na webu, postaveného na OpenAI Assistants API, který by sloužil k okamžitému třídění příchozích dotazů a významně ulehčil práci dispečerů. Dále se doporučuje zavedení voice-bota pro první kontakt využívajícího Twilio a GPT-4o, u kterého se očekává úspora 5–10 minut na každý „teplý“ lead. Významný potenciál má i vytvoření samoobslužného klientského portálu – React aplikace napojené na GPS tracking, která by umožnila cross-sell skladových a celních služeb a mohla by zvýšit zákaznické NPS o 10–12 procentních bodů. Dalším krokem je implementace A/B testování předmětů a CTA pomocí multi-armed bandit algoritmu v platformě Make, kde se plánuje testování 3–5 variant s cílem zvýšit CTR o dalších 1–1,5 procentních bodů. Mezi expanzní strategie patří lokalizace obsahu do jazyků DE/ES/PL a napojení na databáze Kompass a ThomasNet pro vstup na trhy DACH a USA. Dále se doporučuje přejít na personalizovaný LinkedIn outreach zaměřený na 1st-level kontakty (např. SCM Directors), s využitím drip sekvencí tří zpráv a následnými automatickými e-mailovými follow-upy. Posledním klíčovým bodem je integrace CRM (např. HubSpot nebo Pipedrive), která umožní propojit celý akviziční funnel od fáze Sent přes SQL a Quote až po Won. Po třetí realizované přepravě může být navíc automaticky spuštěna NPS anketa.

Implementací navržené AI/RPA architektury LOG-IN CZ převádí tradiční, časově náročnou akvizici do datově řízeného, plně měřitelného procesu. Projekt ukázal, že i středně velká společnost může za omezené náklady dosáhnout metrik, které byly dosud vyhrazeny pouze velkým korporacím s dedikovanými týmy. Díky prokazatelnému snížení nákladů, robustnímu právnímu rámci a jasné strategii škálování je společnost připravena rozšířit inovativní přístup na další trhy a segmenty, čímž si upevní pozici spolehlivého partnera v oblasti vysoce zabezpečené logistiky.

POUŽITÁ LITERATURA

RACKHAM, NEIL. SPIN SELLING, 1988. NEW YORK: MCGRAW-HILL. ISBN 978-0-07-051113-2. [CIT. 2025-02-10].

CHRISTOPHER, MARTIN, 2016. *LOGISTICS AND SUPPLY CHAIN MANAGEMENT*. 5TH ED. HARLOW: PEARSON EDUCATION. ISBN 978-1-292-08379-7. [CIT. 2025-02-20].

STAPLETON, JENNIFER, 1997. DYNAMIC SYSTEMS DEVELOPMENT METHOD: THE METHOD IN PRACTICE. 1ST ED. HARLOW: ADDISON-WESLEY. ISBN 978-0201178890. [CIT. 2025-02-28].

LOG-IN CZ S.R.O, 2024. INTERNÍ DOKUMENTACE OBCHODNÍHO A PROVOZNÍHO ODDĚLENÍ. PARDUBICE. INTERNÍ, NEPUBLIKOVANÝ ZDROJ. [CIT. 2025-02-28].

LOG-IN CZ S.R.O, 2024. INTERNÍ ANALÝZA PROCESŮ SALES & OPERATIONS. PARDUBICE. INTERNÍ, NEPUBLIKOVANÝ ZDROJ. [CIT. 2025-02-28].

RAIN GROUP, 2023. THE 6 ESSENTIAL B2B SALES FUNNEL STAGES [ONLINE]. RAIN GROUP. [CIT. 2025-03-10]. DOSTUPNÉ Z: HTTPS://WWW.RAINGROUP.COM/BLOG/B2B-SALES-FUNNEL-STAGES

REUTERS, 2025. AI-INFLUENCED SHOPPING BOOSTS ONLINE HOLIDAY SALES, SALESFORCE DATA SHOWS [ONLINE]. REUTERS. [CIT. 2025-02-12]. DOSTUPNÉ Z: HTTPS://WWW.REUTERS.COM

MORDOR INTELLIGENCE, 2025. CZECH REPUBLIC FREIGHT & LOGISTICS MARKET – SIZE, SHARE & INDUSTRY TRENDS ANALYSIS [ONLINE]. MORDOR INTELLIGENCE. [CIT. 2025-02-12]. DOSTUPNÉ Z: HTTPS://WWW.MORDORINTELLIGENCE.COM

TECHNAVIO, 2025. ROBOTIC PROCESS AUTOMATION (RPA) MARKET SIZE 2025-2029 [ONLINE]. TECHNAVIO. DOSTUPNÉ Z: HTTPS://WWW.TECHNAVIO.COM

HUBSPOT, 2025. AVERAGE EMAIL OPEN RATES BY INDUSTRY [ONLINE]. HUBSPOT BLOG. [CIT. 2025-02-20]. DOSTUPNÉ Z: HTTPS://BLOG.HUBSPOT.COM/SALES/AVERAGE-EMAIL-OPEN-RATE-BENCHMARK

BUSINESSWIRE, 2025. GENERATIVE ARTIFICIAL INTELLIGENCE IN LOGISTICS – BUSINESS INTELLIGENCE REPORT 2024-2030 [ONLINE]. BUSINESS WIRE. [CIT. 2025-02-17]. DOSTUPNÉ Z: HTTPS://WWW.BUSINESSWIRE.COM

BCG , 2024 AI ADOPTION IN 2024: 74 % OF COMPANIES STRUGGLE TO ACHIEVE SCALE [ONLINE]. BOSTON CONSULTING GROUP. [CIT. 2025-02-15]. DOSTUPNÉ Z: HTTPS://WWW.BCG.COM/PRESS

HUBSPOT, 2023. EMAIL MARKETING BENCHMARKS [ONLINE]. HUBSPOT. [CIT. 2025-02-28]. DOSTUPNÉ Z: HTTPS://BLOG.HUBSPOT.COM/MARKETING/EMAIL-MARKETING-BENCHMARKS

MORDOR INTELLIGENCE, 2025. GENERATIVE AI IN LOGISTICS MARKET [ONLINE]. 2025. MORDOR INTELLIGENCE. [CIT. 2025-03-17].DOSTUPNÉ Z: HTTPS://WWW.MORDORINTELLIGENCE.COM

EUROPEAN COMMISSION, 2024. USE OF ARTIFICIAL INTELLIGENCE IN ENTERPRISES – STATISTICS EXPLAINED [ONLINE]. EUROSTAT. [CIT. 2025-02-17]. DOSTUPNÉ Z: HTTPS://EC.EUROPA.EU/EUROSTAT/STATISTICS-EXPLAINED

ÚOOÚ, 2024. OBECNĚ K ZÁKONU Č. 480/2004 SB. – NEVYŽÁDANÁ OBCHODNÍ SDĚLENÍ [ONLINE]. ÚŘAD PRO OCHRANU OSOBNÍCH ÚDAJŮ. [CIT. 2025-02-19]. DOSTUPNÉ Z: HTTPS://UOOU.GOV.CZ

EDPB, 2024. GUIDELINES 1/2024 ON PROCESSING OF PERSONAL DATA FOR LEGITIMATE INTERESTS PURPOSES [ONLINE]. EUROPEAN DATA PROTECTION BOARD. [CIT. 2025-03-17]. DOSTUPNÉ Z: HTTPS://EDPB.EUROPA.EU

GLOBAL PRIVACY BLOG, 2024. EDPB ISSUES GUIDELINES ON PROCESSING PERSONAL DATA [ONLINE]. HUNTON ANDREWS KURTH LLP. [CIT. 2025-03-17]. DOSTUPNÉ Z: HTTPS://WWW.GLOBALPRIVACYBLOG.COM

EUROPEAN COMMISSION, 2024. GENERAL DATA PROTECTION REGULATION (GDPR) COMPLIANCE GUIDELINES [ONLINE]. EUROPEAN COMMISSION. [CIT. 2025-02-19]. DOSTUPNÉ Z: HTTPS://COMMISSION.EUROPA.EU/LAW/GDPR_EN

EDPB, 2024. GUIDELINES 1/2024 ON THE PROCESSING OF PERSONAL DATA FOR DIRECT MARKETING PURPOSES [ONLINE]. EUROPEAN DATA PROTECTION BOARD. [CIT. 2025-03-19]. DOSTUPNÉ Z: HTTPS://EDPB.EUROPA.EU

HUBSPOT, 2024. TOP DIGITAL MARKETING TRENDS IN 2025 [ONLINE]. HUBSPOT BLOG. [CIT. 2025-04-17]. DOSTUPNÉ Z: HTTPS://BLOG.HUBSPOT.COM/MARKETING/HUBSPOT-BLOG-MARKETING-INDUSTRY-TRENDS-REPORT?HUBS_CONTENT=

MAKE, 2024. AUTOMATION SOFTWARE | CONNECT APPS & DESIGN WORKFLOWS [ONLINE]. MAKE.COM. [CIT. 2025-03-10]. DOSTUPNÉ Z: HTTPS://WWW.MAKE.COM

APIFY, 2024. GOOGLE MAPS SCRAPER – APIFY [ONLINE]. APIFY. [CIT. 2025-02-17]. DOSTUPNÉ Z: HTTPS://APIFY.COM/COMPASS/CRAWLER-GOOGLE-PLACES

OPENAI, 2025. INTRODUCING GPT-4.1 IN THE API [ONLINE]. OPENAI. [CIT. 2025-03-17]. DOSTUPNÉ Z: HTTPS://OPENAI.COM

GOOGLE, 2024. GOOGLE WORKSPACE ADMIN HELP – SPF, DKIM, DMARC SETTINGS [ONLINE]. GOOGLE. [CIT. 2025-03-19]. DOSTUPNÉ Z: HTTPS://SUPPORT.GOOGLE.COM/SEARCH?Q=SPF%2C+DKIM%2C+DMARC+SETTINGS+&FROM_PROMOTED_SEARCH=TRUE

HUBSPOT, 2024. THE STATE OF AI IN BUSINESS AND SALES – NOV 2024 DATA & STATISTICS [ONLINE]. HUBSPOT BLOG. [CIT. 2025-04-19]. DOSTUPNÉ Z: HTTPS://BLOG.HUBSPOT.COM/SALES/STATE-OF-AI-SALES

COMPLIANCE HUB, 2024. TOP GDPR FINES IN DECEMBER 2024: KEY LESSONS FOR COMPLIANCE [ONLINE]. COMPLIANCE HUB WIKI. DOSTUPNÉ Z: HTTPS://WWW.COMPLIANCEHUB.WIKI

ISO, 2024. CERTIFIED CLIENTS - ISO 9001 [ONLINE]. DOSTUPNÉ Z: HTTPS://WWW.ISO.ORG

CHATGPT, 2024. CHATGPT - LARGE LANGUAGE MODEL INTERFACE [ONLINE]. OPENAI. DOSTUPNÉ Z: HTTPS://CHATGPT.COM .

COURSERA, 2025. SUPERVISED MACHINE LEARNING: REGRESSION AND CLASSIFICATION [ONLINE]. DOSTUPNÉ Z: HTTPS://WWW.COURSERA.ORG/LEARN/MACHINE-LEARNING

DUN & BRADSTREET, 2024. HODNOCENÍ FIREM: LOG-IN CZ S.R.O. [ONLINE]. DOSTUPNÉ Z: HTTPS://WWW.DNB.COM

PRECEDENCE RESEARCH, 2025. ROBOTIC PROCESS AUTOMATION MARKET SIZE AND FORECAST 2025-2034 [ONLINE]. PRECEDENCE RESEARCH. DOSTUPNÉ Z: HTTPS://WWW.PRECEDENCERESEARCH.COM

GETRESPONSE, 2024. EMAIL MARKETING BENCHMARKS & STATISTICS (UPDATED FOR 2024) [ONLINE]. GETRESPONSE. [CIT. 2025-05-14]. DOSTUPNÉ Z: HTTPS://WWW.GETRESPONSE.COM/RESOURCES/REPORTS/EMAIL-MARKETING-BENCHMARKS

RESEARCHANDMARKETS, 2024. GENERATIVE ARTIFICIAL INTELLIGENCE IN LOGISTICS [ONLINE]. RESEARCHANDMARKETS. DOSTUPNÉ Z: HTTPS://WWW.RESEARCHANDMARKETS.COM

TOM'S GUIDE, 2025. STOP WASTING TIME WITH THE WRONG CHATGPT MODEL – HERE'S HOW TO CHOOSE THE RIGHT ONE [ONLINE]. TOM'S GUIDE. DOSTUPNÉ Z: HTTPS://WWW.TOMSGUIDE.COM

GOHIGHLEVEL, 2024. EMAIL SENDING GUIDE: EMAIL BEST PRACTICES & EMAIL WARM UP [ONLINE]. HIGHLEVEL SUPPORT PORTAL. DOSTUPNÉ Z: HTTPS://HELP.GOHIGHLEVEL.COM

SEZNAM TABULEK

SEZNAM OBRÁZKŮ

SEZNAM ZKRATEK

2FA	Dvoufaktorové ověřování (Two-Factor Authentication)
AEO	Oprávněný hospodářský subjekt
AI	Umělá inteligence (Artificial Intelligence)
API	Aplikační programové rozhraní
APAC	Region Asie a Tichomoří (Asia-Pacific)
BAT V1	Business Approval Tool, verze 1
BI	Obchodní analytika (Business Intelligence)
BQ	BigQuery – cloudové úložiště dat Google
B2B	Obchod mezi firmami (Business-to-Business)
CAPEX	Kapitálové výdaje (Capital Expenditures)
CAGR	Složená roční míra růstu (Compound Annual Growth Rate)
CEO	Generální ředitel (Chief Executive Officer)
CLV / LTV	Životní hodnota zákazníka (Customer/Lifetime Value)
CRM	Řízení vztahů se zákazníky
CTR	Míra prokliku (Click-Through Rate)
CSV	Textový formát hodnot oddělených čárkou
CTA	Výzva k akci (Call-to-Action)
DC	Datové centrum (Data Center)
DKIM	Digitální podpis e-mailu (DomainKeys Identified Mail)
DNS	Systém doménových jmen (Domain Name System)
DPA	Smlouva o zpracování údajů (Data Processing Agreement)
DPO	Pověřenec pro ochranu osobních údajů (Data Protection Officer)
EDI	Elektronická výměna dat (Electronic Data Interchange)
EDPB	Evropský sbor pro ochranu osobních údajů
EU	Evropská unie
ESP	Poskytovatel e-mailových služeb *(Email Service Provider)*
FTL	Celovozová přeprava (Full Truck Load)
GDPR	Obecné nařízení o ochraně osobních údajů
GUI	Grafické uživatelské rozhraní (Graphical User Interface)
HR	Lidské zdroje (Human Resources)
HTML	Hypertextový značkovací jazyk (HyperText Markup Language)
ID	Identifikátor (Identifier)
IT	Informační technologie

JS	JavaScript – skriptovací jazyk webu
KPI	Klíčový ukazatel výkonnosti (Key Performance Indicator)
LATAM	Latinská Amerika (Latin America)
LIA	Posouzení oprávněného zájmu (Legitimate Interest Assessment)
LLM	Velký jazykový model (Large Language Model)
LTL	Částečná přeprava (Less Than Truck Load)
ML	Strojové učení (Machine Learning)
MX	Mail-Exchange záznam v DNS
NPS	Čisté skóre podpory (Net Promoter Score)
OAuth2	Protokol otevřeného ověřování 2.0 (Open Authorization 2.0)
OPEX	Provozní náklady (Operating Expenditures)
PDF	Přenositelný formát dokumentů (Portable Document Format)
RPA	Robotická procesní automatizace (Robotic Process Automation)
ROI	Návratnost investice (Return on Investment)
SCC	Standardní smluvní doložky (Standard Contractual Clauses)
SLA	Dohoda o úrovni služby (Service Level Agreement)
SMTP	Protokol pro odesílání e-mailů (Simple Mail Transfer Protocol)
SME	Malé a střední podniky (Small and Medium-sized Enterprises)
SOC 2	Bezpečnostní auditní standard
SQL (lead)	Obchodně kvalifikovaný lead (Sales Qualified Lead)
SWOT	Analýza silných a slabých stránek, příležitostí a hrozeb
SEO	Optimalizace pro vyhledávače *(Search Engine Optimization)*
TAPA	Asociace pro ochranu přepravovaného majetku
TSR 1	Požadavky na zabezpečení přepravy, úroveň 1
TLS	Bezpečnost transportní vrstvy (Transport Layer Security)
URL	Jednotný lokátor zdroje (Uniform Resource Locator)
UTM	Parametry pro sledování kampaní (Urchin Tracking Module)
VPS	Virtuální privátní server (Virtual Private Server)
vCPU	Virtualizované jádro procesoru (Virtual CPU)
WWW	Světová síť (World Wide Web)

SEZNAM PŘÍLOH

Příloha A bot.py

Příloha B my_custom_parser.py

Příloha C Search_Gmail.py

Příloha D linkedin_sender.py

Příloha E facebook_sender.py

Příloha F profiles.txt

Příloha G email_validator.py

Příloha H Legitimate Interest Assessment (LIA) LOG-IN

Příloha I setup_server.sh

Příloha A bot.py

```
import logging
import requests
import json
from telegram import Update, InlineKeyboardButton, InlineKeyboardMarkup
from telegram.ext import (
    Application,
    ApplicationBuilder,
    CommandHandler,
    CallbackQueryHandler,
    MessageHandler,
    ContextTypes,
    filters
)
from selenium.webdriver.common.by import By
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium import webdriver

# ------------------ НАСТРОЙКИ И КОНСТАНТЫ ------------------

CHROMEDRIVER_PATH = "/Users/user/Desktop/Log IN/chromedriver"

# Вебхуки (можно хранить в .env-файле или другом месте)
WEBHOOK_URL_SWOT = "https://hook.eu2.make.com/1fupvg6dpklbjz5u9sg7jiqnsrxgvlca"
WEBHOOK_URL_NEW_CLIENTS = "https://hook.eu2.make.com/1fupvg6dpklbjz5u9sg7jiqnsrxgvlca"
WEBHOOK_URL_SITE_ANALYSIS = "https://hook.eu2.make.com/1fupvg6dpklbjz5u9sg7jiqnsrxgvlca"
WEBHOOK_URL_CUSTOMER_RESPONSE = "https://hook.eu2.make.com/1fupvg6dpklbjz5u9sg7jiqnsrxgvlca"
WEBHOOK_URL_GOOGLE_SEARCH = "https://hook.eu2.make.com/1fupvg6dpklbjz5u9sg7jiqnsrxgvlca"
WEBHOOK_URL_TEXT_PIC = "https://hook.eu2.make.com/1fupvg6dpklbjz5u9sg7jiqnsrxgvlca"

# Ваш токен бота (рекомендуется хранить его в безопасном месте!)
BOT_TOKEN = "7612407473:AAFtdMr18KyI_-7mX0J9XOUWx7oE_ciYh94"

# Логирование [redacted]
logging.basicConfig(
    format='%(asctime)s - %(name)s - %(levelname)s - %(message)s',
    level=logging.INFO
)
logger = logging.getLogger(__name__)

# Глобальные словари для хранения промежуточных данных
user_data = {}
message_ids = {}
processed_updates = set()  # Хранение обработанных update_id (чтобы не дублировать)

# ------------------ ВСПОМОГАТЕЛЬНЫЕ ФУНКЦИИ ------------------

async def send_json_to_webhook(url: str, payload: dict) -> (bool, str):
    """
    Универсальная функция отправки JSON на указанный вебхук.
    Возвращает (успешно_ли, текст_ответа).
    """
    try:
        response = requests.post(url, json=payload, timeout=90)
        return (response.status_code == 200, response.text)
    except Exception as e:
        logger.exception("Ошибка при отправке JSON на вебхук")
        return (False, str(e))

# ------------------ ОБРАБОТЧИКИ КОМАНД ------------------

async def start(update: Update, context: ContextTypes.DEFAULT_TYPE):
    """
    Команда /start — выводит меню с кнопками.
    """
    user_id = update.effective_user.id
    update_id = update.update_id



    # Чтобы /start не вызывался повторно для одного update_id
    if update_id in processed_updates:
        logger.info(f"/start уже был вызван (update_id={update_id}). Пропускаем.")
        return
    processed_updates.add(update_id)

    keyboard = [
        [InlineKeyboardButton("📊 SWOT analýza", callback_data='swot')],
        [InlineKeyboardButton("🗺️ Hledat nové klienty Google Maps", callback_data='new_clients')],
        [InlineKeyboardButton("🔧 Vytvoření jedinečné nabídky", callback_data='unique_offer')],
        [InlineKeyboardButton("✉️ Odpověď na dopis zákazníka", callback_data='customer_response')],
        [InlineKeyboardButton("🌐 Noví klienty Google search", callback_data='google_search')],
        [InlineKeyboardButton("🔍 Vyhledat na Europages", callback_data='europages')],
        [InlineKeyboardButton("📝 Z fotografie do textu", callback_data='photo_to_text')],
        [InlineKeyboardButton("🗑️ Smazat chat", callback_data='delete_chat')]
    ]
    reply_markup = InlineKeyboardMarkup(keyboard)

    message = await update.message.reply_text(
        "Vyberte akci:",
        reply_markup=reply_markup
    )
    message_ids.setdefault(user_id, []).append(message.message_id)

async def command1(update: Update, context: ContextTypes.DEFAULT_TYPE, from_button=False):
    """
    SWOT analýza
    """
    user_id = update.effective_user.id
    user_data[user_id] = {'action': 'SWOT analýza', 'step': 'collect_query'}

    text_prompt = (
        "Zadejte oblast nebo typ podnikání, pro který chcete provést SWOT analýzu "
        "(např. logistická společnost, online obchod, výrobní firma apod.):"
    )

    if from_button:
        message = await update.callback_query.message.reply_text(text_prompt)
    else:
        message = await update.message.reply_text(text_prompt)

    message_ids.setdefault(user_id, []).append(message.message_id)

async def command2(update: Update, context: ContextTypes.DEFAULT_TYPE, from_button=False):
    """
    Noví klienti
    """
    user_id = update.effective_user.id
    user_data[user_id] = {'action': 'Noví klienti', 'step': 'collect_query'}

    text_prompt = (
        "Zadejte kategorii klientů, pro které hledáte nové příležitosti "
        "(např. výrobní společnosti, obchodní firmy, luxusní zboží apod.):"
    )

    if from_button:
        message = await update.callback_query.message.reply_text(text_prompt)
    else:
        message = await update.message.reply_text(text_prompt)

    message_ids.setdefault(user_id, []).append(message.message_id)

async def command3(update: Update, context: ContextTypes.DEFAULT_TYPE, from_button=False):
    """
    Vytvoření jedinečné nabídky
    """
    user_id = update.effective_user.id
```

```
    user_id = update.effective_user.id
    user_data[user_id] = {'action': 'Vytvoření jedineční nabídky', 'step': 'collect_link'}

    text_prompt = "Zadejte prosím odkaz na webovou stránku pro vytvoření jedineční nabídky:"

    if from_button:
        message = await update.callback_query.message.reply_text(text_prompt)
    else:
        message = await update.message.reply_text(text_prompt)

    message_ids.setdefault(user_id, []).append(message.message_id)

async def command4(update: Update, context: ContextTypes.DEFAULT_TYPE, from_button=False):
    """
    Odpověď na dopis zákazníka
    """
    user_id = update.effective_user.id
    user_data[user_id] = {'action': 'Odpověď na dopis zákazníka', 'step': 'collect_email'}

    text_prompt = "Zadejte prosím Gmail zákazníka:"

    if from_button:
        message = await update.callback_query.message.reply_text(text_prompt)
    else:
        message = await update.message.reply_text(text_prompt)

    message_ids.setdefault(user_id, []).append(message.message_id)

async def command5(update: Update, context: ContextTypes.DEFAULT_TYPE, from_button=False):
    """
    Noví klienty Google search
    """
    user_id = update.effective_user.id
    user_data[user_id] = {'action': 'Noví klienty Google search', 'step': 'ask_site_search'}

    keyboard = [
        [InlineKeyboardButton("Hledat na webu firmy.cz", callback_data="firmy")],
        [InlineKeyboardButton("Hledat na Insagram", callback_data="Instagram")],
        [InlineKeyboardButton("Hledat na LinkedIn", callback_data="LinkedIn")],
        [InlineKeyboardButton("Hledat na Facebook", callback_data="Facebook")],
        [InlineKeyboardButton("Hledat na celém internetu", callback_data="internet")]
    ]
    reply_markup = InlineKeyboardMarkup(keyboard)

    text_prompt = "Kde chcete hledat?"

    if from_button:
        message = await update.callback_query.message.reply_text(text_prompt, reply_markup=reply_markup)
    else:
        message = await update.message.reply_text(text_prompt, reply_markup=reply_markup)

    message_ids.setdefault(user_id, []).append(message.message_id)

async def command6(update: Update, context: ContextTypes.DEFAULT_TYPE, from_button=False):
    """
    Odeslání fotografie
    """
    user_id = update.effective_user.id
    user_data[user_id] = {'action': 'Odeslání fotografie', 'step': 'collect_photo'}

    text_prompt = "Zašlete prosím obrázek:"

    if from_button:
        message = await update.callback_query.message.reply_text(text_prompt)
    else:
        message = await update.message.reply_text(text_prompt)

    message_ids.setdefault(user_id, []).append(message.message_id)



    message_ids.setdefault(user_id, []).append(message.message_id)

# ------------------ ОБРАБОТКА КНОПОК ------------------

async def button_handler(update: Update, context: ContextTypes.DEFAULT_TYPE):
    query = update.callback_query
    await query.answer()
    user_id = query.from_user.id
    data = query.data

    # В зависимости от callback_data (query.data) вызываем нужную команду или логику
    if data == 'swot':
        await command1(update, context, from_button=True)
    elif data == 'new_clients':
        await command2(update, context, from_button=True)
    elif data == 'unique_offer':
        await command3(update, context, from_button=True)
    elif data == 'customer_response':
        await command4(update, context, from_button=True)
    elif data == 'google_search':
        await command5(update, context, from_button=True)
    elif data == 'europages':
        await europages_button_handler(update, context, from_button=True)

    elif data in ('firmy', 'Instagram', 'LinkedIn', 'Facebook', 'internet'):
        site_map = {
            'firmy': ("https://www.firmy.cz/", "i"),
            'Instagram': ("https://www.instagram.com", "i"),
            'LinkedIn': ("https://www.linkedin.com", "i"),
            'Facebook': ("https://www.facebook.com", "i"),
            'internet': ("", "")
        }
        siteSearch, siteSearchFilter = site_map[data]
        user_data[user_id]['siteSearch'] = siteSearch
        user_data[user_id]['siteSearchFilter'] = siteSearchFilter
        user_data[user_id]['step'] = 'collect_keywords'

        text_prompt = "Zadejte klíčová slova pro vyhledávání klientů:"
        message = await query.message.reply_text(text_prompt)
        message_ids.setdefault(user_id, []).append(message.message_id)

    elif data == 'photo_to_text':
        await command6(update, context, from_button=True)
    elif data == 'delete_chat':
        await delete_chat(update, context)

# ------------------ ОБРАБОТКА ТЕКСТОВ ------------------

async def text_handler(update: Update, context: ContextTypes.DEFAULT_TYPE):
    user_id = update.message.from_user.id

    # Если пользователь неизвестен — просим ввести /start
    if user_id not in user_data:
        await update.message.reply_text("Začněte prosím s příkazem /start.")
        return

    current_step = user_data[user_id].get('step')
    current_action = user_data[user_id].get('action')

    # ------------------ ОБРАБОТКА EUROPAGES ------------------
    if current_action == 'europages':
        # 1. Пользователь вводит поисковой запрос
        if current_step == 'collect_europages_query':
            user_data[user_id]['query'] = update.message.text
            user_data[user_id]['step'] = 'collect_europages_country'
            await update.message.reply_text("Zadejte prosím zemi, ve které chcete hledat:")
            return
```

```
        user_data[user_id]['step'] = 'collect_europages_count'
        await update.message.reply_text("Zadejte počet společností, které chcete vyhledat (např. 5):")
        return

    # 3. Пользователь указывает, сколько компаний получить
    elif current_step == 'collect_europages_count':
        try:
            num_results = int(update.message.text)
            user_data[user_id]['count'] = num_results

            user_data[user_id]['step'] = 'collect_europages_start_page'
            await update.message.reply_text("Zadejte číslo stránky, od které chcete začít vyhledávat (např. 1)
        except ValueError:
            await update.message.reply_text("Zadejte prosím správné číslo.")
        return

    # 4. Пользователь указывает, с какой страницы начать
    elif current_step == 'collect_europages_start_page':
        try:
            start_page = int(update.message.text)
            user_data[user_id]['start_page'] = start_page

            # Данные готовы для запуска парсинга
            query_str = user_data[user_id]['query']
            country_str = user_data[user_id]['country']
            num_results = user_data[user_id]['count']
            start_pg = user_data[user_id]['start_page']

            await update.message.reply_text(
                f"Hledám {num_results} společností pro dotaz '{query_str}' "
                f"v zemi '{country_str}' na stránce {start_pg} na Europages..."
            )

            # Импортируем ваш парсер (скрипт для Europages).
            # Допустим, в модуле my_custom_parser у вас есть функция scrape_europages(...)
            from my_custom_parser import scrape_europages

            # Запускаем парсер с учётом введённой страны и start_page
            website_links, total_companies = scrape_europages(
                query=query_str,
                country=country_str,
                num_results=num_results,
                start_page=start_pg  # <-- ВАЖНО: передаём start_page
            )

            # Отправляем результат на вебхук
            payload = {
                "source": "europages",
                "query": query_str,
                "country": country_str,
                "requested_count": num_results,
                "start_page": start_pg,
                "total_companies_found": total_companies,
                "links": website_links
            }
            success, resp_text = await send_json_to_webhook(WEBHOOK_URL_GOOGLE_SEARCH, payload)
            if success:
                logger.info("Результат Europages успешно отправлен на вебхук.")
            else:
                logger.warning(f"Не удалось отправить на вебхук: {resp_text}")

            # Можно отправить ссылку на Google-таблицу (пример):
            await update.message.reply_text(
                "Všechny výsledky jsou v tabulce:\n"
                "https://docs.google.com/spreadsheets/d/1LWObTdS3nKchtT5P1QDTT4HYIj0WbkkGU4edavEg7NQ/edit?gid=
            )

        except ValueError:



            )

        except ValueError:
            await update.message.reply_text("Zadejte prosím správné číslo stránky.")
        except Exception as e:
            logger.exception("Ошибка при parsu Europages")
            await update.message.reply_text(f"Došlo k chybě: {e}")
        finally:
            # Сбрасываем данные
            user_data.pop(user_id, None)
        return

# ------------------ ПРОЧИЕ ДЕЙСТВИЯ (SWOT, NOVÍ KLIENTI и т.д.) ------------------

if current_step == 'collect_query':
    user_data[user_id]['query'] = update.message.text
    user_data[user_id]['step'] = 'collect_count'
    message = await update.message.reply_text("Zadejte počet dotazů k hledání:")
    message_ids[user_id].append(message.message_id)

elif current_step == 'collect_count':
    try:
        user_data[user_id]['count'] = int(update.message.text)
        user_data[user_id]['step'] = 'collect_country'
        message = await update.message.reply_text("Zadejte zemi:")
        message_ids[user_id].append(message.message_id)
    except ValueError:
        message = await update.message.reply_text("Zadejte prosím číslo.")
        message_ids[user_id].append(message.message_id)

elif current_step == 'collect_country':
    user_data[user_id]['country'] = update.message.text
    user_data[user_id]['step'] = 'collect_city'
    message = await update.message.reply_text("Zadejte město:")
    message_ids[user_id].append(message.message_id)

elif current_step == 'collect_city':
    user_data[user_id]['city'] = update.message.text
    user_data[user_id]['step'] = 'ready'
    await send_to_webhook(update, context)

elif current_step == 'collect_keywords':
    user_data[user_id]['keywords'] = update.message.text
    user_data[user_id]['step'] = 'collect_city_google'
    message = await update.message.reply_text("Zadejte město nebo zemi:")
    message_ids[user_id].append(message.message_id)

elif current_step == 'collect_city_google':
    user_data[user_id]['city'] = update.message.text
    user_data[user_id]['step'] = 'collect_start'
    message = await update.message.reply_text("Zadejte číslo stránky, na které chcete hledat výsledky:")
    message_ids[user_id].append(message.message_id)

elif current_step == 'collect_start':
    try:
        user_data[user_id]['start'] = int(update.message.text)
        user_data[user_id]['step'] = 'collect_num'
        message = await update.message.reply_text("Zadejte počet výsledků, které chcete zobrazit (1–100):")
        message_ids[user_id].append(message.message_id)
    except ValueError:
        message = await update.message.reply_text("Prosím, zadejte platné číslo stránky.")
        message_ids[user_id].append(message.message_id)

elif current_step == 'collect_num':
    try:
        user_data[user_id]['num'] = int(update.message.text)
        user_data[user_id]['step'] = 'ready'
        await send_to_webhook(update, context)
```

```
        user_data[user_id]['step'] = 'ready'
        await send_to_webhook(update, context)

    elif current_step == 'collect_email':
        user_data[user_id]['email'] = update.message.text
        user_data[user_id]['step'] = 'collect_message'
        message = await update.message.reply_text("Zadejte prosím zprávu zákazníkovi:")
        message_ids[user_id].append(message.message_id)

    elif current_step == 'collect_message':
        user_data[user_id]['message'] = update.message.text
        user_data[user_id]['step'] = 'ready'
        await send_to_webhook(update, context)

# ------------------ УДАЛЕНИЕ ЧАТА ------------------

async def delete_chat(update: Update, context: ContextTypes.DEFAULT_TYPE):
    """
    Команда /delete_chat — удаляет все сообщения, сохранённые для данного пользователя
    """
    user_id = update.effective_user.id
    chat_id = update.effective_chat.id

    if user_id not in message_ids or not message_ids[user_id]:
        await update.message.reply_text("Žádné zprávy k odstranění.")
        return

    for mid in message_ids[user_id]:
        try:
            await context.bot.delete_message(chat_id=chat_id, message_id=mid)
        except Exception as e:
            logger.error(f"Не удалось удалить сообщение {mid}: {e}")

    message_ids[user_id] = []
    confirmation = await update.message.reply_text("Chat byl vymazán. Začněte znovu pomocí /start.")
    message_ids[user_id].append(confirmation.message_id)

# ------------------ ОТПРАВКА НА ВЕБХУК (общая) ------------------

async def send_to_webhook(update: Update, context: ContextTypes.DEFAULT_TYPE):
    """
    Отправляем собранные данные на нужный вебхук (в зависимости от действия).
    """
    user_id = update.message.from_user.id
    data = user_data[user_id]

    action = data['action']
    if action == 'SWOT analýza':
        webhook_url = WEBHOOK_URL_SWOT
    elif action == 'Noví klienti':
        webhook_url = WEBHOOK_URL_NEW_CLIENTS
    elif action == 'Vytvoření jedineční nabídky':
        webhook_url = WEBHOOK_URL_SITE_ANALYSIS
    elif action == 'Odpověď na dopis zákazníka':
        webhook_url = WEBHOOK_URL_CUSTOMER_RESPONSE
    else:
        webhook_url = WEBHOOK_URL_GOOGLE_SEARCH

    msg = await update.message.reply_text("Počkejte prosím, váš dotaz se zpracovává...")
    message_ids[user_id].append(msg.message_id)

    payload = {
        "user": update.message.from_user.username,
        "action": action,
        "query": data.get('query'),
        "count": data.get('count'),
        "country": data.get('country'),
        "citv": data.get('citv').



        "email": data.get('email'),
        "message": data.get('message'),
        "keywords": data.get('keywords'),
        "start": data.get('start'),
        "num": data.get('num'),
        "siteSearch": data.get('siteSearch'),
        "siteSearchFilter": data.get('siteSearchFilter'),
    }

    success, resp_text = await send_json_to_webhook(webhook_url, payload)
    if success:
        logger.info(f"Успешно отправлено на {webhook_url}")
    else:
        logger.warning(f"Не удалось отправить на вебхук: {resp_text}")

    # Предполагаем, что вебхук возвращает JSON
    # Попробуем его разобрать
    try:
        result = json.loads(resp_text)
        # Логика ответа
        if 'message pic' in result:
            msg2 = await update.message.reply_text(result['message pic'])
            message_ids[user_id].append(msg2.message_id)
        elif 'file_url' in result:
            msg2 = await update.message.reply_text(f"Zde je váš odkaz na soubor: {result['file_url']}")
            message_ids[user_id].append(msg2.message_id)
        elif 'message' in result:
            msg2 = await update.message.reply_text(result['message'])
            message_ids[user_id].append(msg2.message_id)
        else:
            logger.error(f"Nesprávná odpověď: {resp_text}")
            msg2 = await update.message.reply_text(f"Chyba: nesprávná odpověď serveru: {resp_text}")
            message_ids[user_id].append(msg2.message_id)
    except json.JSONDecodeError:
        # Если это не JSON
        logger.error(f"Сервер вернул не JSON: {resp_text}")
        msg2 = await update.message.reply_text(f"Chyba: server vrátil neplatný JSON. Odpověď: {resp_text}")
        message_ids[user_id].append(msg2.message_id)

    # Очищаем данные пользователя
    user_data.pop(user_id, None)

# ------------------ ОБРАБОТКА ФОТОГРАФИЙ ------------------

async def photo_handler(update: Update, context: ContextTypes.DEFAULT_TYPE):
    """
    Обрабатывает входящие фотографии, если пользователь находится в шаге collect_photo.
    """
    user_id = update.message.from_user.id
    if user_id not in user_data or user_data[user_id].get('step') != 'collect_photo':
        msg = await update.message.reply_text("Začněte příkazem /start.")
        message_ids.setdefault(user_id, []).append(msg.message_id)
        return

    # Получаем фото
    photo_file = await update.message.photo[-1].get_file()
    photo_bytes = await photo_file.download_as_bytearray()

    user_data[user_id]['photo'] = photo_bytes
    user_data[user_id]['step'] = 'ready'
    await send_photo_to_webhook(update, context)

async def send_photo_to_webhook(update: Update, context: ContextTypes.DEFAULT_TYPE):
    """
    Отправляем фотографию на соответствующий вебхук
    """
    user_id = update.message.from_user.id
    data = user_data[user_id]
```

```
    msg = await update.message.reply_text("Zpracování fotografií. Počkejte prosím...")
    message_ids[user_id].append(msg.message_id)

    try:
        files = {"photo": ("photo.jpg", data['photo'], "image/jpeg")}
        form_data = {
            "user": update.message.from_user.username,
            "action": data['action']
        }
        response = requests.post(WEBHOOK_URL_TEXT_PIC, files=files, data=form_data, timeout=30)

        resp_text = response.text
        try:
            result = json.loads(resp_text)
            if 'message pic' in result:
                msg2 = await update.message.reply_text(result['message pic'])
                message_ids[user_id].append(msg2.message_id)
            else:
                msg2 = await update.message.reply_text("Chyba: nesprávná odpověď serveru.")
                message_ids[user_id].append(msg2.message_id)
        except json.JSONDecodeError:
            msg2 = await update.message.reply_text(
                f"Chyba: server vrátil neplatný JSON. Odpověď: {resp_text}"
            )
            message_ids[user_id].append(msg2.message_id)

    except Exception as e:
        logger.exception("Ошибка при отправке фото на webhook")
        msg2 = await update.message.reply_text(f"Došlo k chybě: {e}")
        message_ids[user_id].append(msg2.message_id)

    user_data.pop(user_id, None)

# ------------------ ОБРАБОТЧИК ДЛЯ /europages ------------------

async def europages_button_handler(update: Update, context: ContextTypes.DEFAULT_TYPE, from_button=False):
    """
    Регистрируем действие 'europages' и шаг 'collect_europages_query'.
    """
    user_id = update.effective_user.id if not from_button else update.callback_query.from_user.id
    user_data[user_id] = {'action': 'europages', 'step': 'collect_europages_query'}

    text_prompt = "Zadejte dotaz pro vyhledávání společností na stránkách Europages:"

    if from_button:
        msg = await update.callback_query.message.reply_text(text_prompt)
    else:
        msg = await update.message.reply_text(text_prompt)

    message_ids.setdefault(user_id, []).append(msg.message_id)

# ------------------ ОСНОВНАЯ ФУНКЦИЯ ------------------

def main():
    """
    Точка входа в приложение. Регистрируем все обработчики и запускаем бота.
    """
    application = Application.builder().token(BOT_TOKEN).build()

    # Регистрируем команды
    application.add_handler(CommandHandler("start", start))
    application.add_handler(CommandHandler("command1", command1))
    application.add_handler(CommandHandler("command2", command2))
    application.add_handler(CommandHandler("command3", command3))
    application.add_handler(CommandHandler("command4", command4))
    application.add_handler(CommandHandler("command5", command5))
    application.add_handler(CommandHandler("command6", command6))


    application.add_handler(CommandHandler("europages", europages_button_handler))
    application.add_handler(CommandHandler("delete_chat", delete_chat))

    # Регистрируем обработчик нажатия кнопок (CallbackQuery)
    application.add_handler(CallbackQueryHandler(button_handler))

    # Регистрируем обработчик фотографий
    application.add_handler(MessageHandler(filters.PHOTO, photo_handler))

    # Регистрируем обработчик текста (не команд)
    application.add_handler(MessageHandler(filters.TEXT & ~filters.COMMAND, text_handler))

    # Запускаем бота (long polling)
    application.run_polling()

if __name__ == '__main__':
    main()
```

Zdroj: Autor

Příloha B my_custom_parser.py

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.common.keys import Keys
from selenium.common.exceptions import (
    TimeoutException,
    StaleElementReferenceException,
    NoSuchElementException
)
import requests
import time

CHROMEDRIVER_PATH = "/Users/user/Desktop/Log IN/chromedriver"


def accept_cookies(driver, timeout=20):
    """
    Принимает cookies на сайте Europages, если появляется окно.
    """
    try:
        cookies_button = WebDriverWait(driver, timeout).until(
            EC.element_to_be_clickable((By.ID, "CybotCookiebotDialogBodyLevelButtonLevelOptinAllowAll"))
        )
        cookies_button.click()
        print("[INFO] Cookies byly úspěšně přijaty.")
    except TimeoutException:
        print("[INFO] Окно с cookies либо не появилось, либо уже было закрыто.")


def select_country_location(driver, country, timeout=20):
    """
    В поле «location» вводит название страны, жмёт ENTER,
    после чего кнопка 'Použít' становится активной и мы на неё кликаем.
    """
    try:
        # Ищем поле ввода локации
        location_input = WebDriverWait(driver, timeout).until(
            EC.element_to_be_clickable((By.CSS_SELECTOR, "input#location"))
        )
        location_input.clear()
        location_input.send_keys(country)
        print(f"[INFO] Ввели страну в поле: {country}")

        time.sleep(1)  # Пауза, чтобы автокомплит успел среагировать

        # ENTER
        location_input.send_keys(Keys.ENTER)
        print("[INFO] Нажали ENTER в поле 'location'.")
        time.sleep(2)

        # Кликаем «Použít»
        apply_button = WebDriverWait(driver, timeout).until(
            EC.element_to_be_clickable((By.CSS_SELECTOR, "button[data-test='apply-location']"))
        )
        apply_button.click()
        print("[INFO] Кликнули на кнопку 'Použít'.")
        time.sleep(3)
    except TimeoutException:
        print("[ERROR] Не удалось задать локацию или кликнуть на 'Použít'.")

def extract_website_link(driver, company_url, timeout=15):
    """
    Извлекает (если есть) ссылку на веб-сайт компании со страницы компании.
    Возвращает str (URL) или None.
    """


    """
    try:
        # Ищем <span> внутри кнопки «Navštívit webové stránky» (по конкретному селектору)
        website_span = WebDriverWait(driver, timeout).until(
            EC.presence_of_element_located((
                By.CSS_SELECTOR,
                "#__nuxt > div > div:nth-child(1) > div:nth-child(2) > div > "
                "div.company-profile.flex.flex-col.gap-2.py-2.lg\\:flex-row > "
                "div.flex.flex-col.gap-2.lg\\:min-w-\\[250px\\].lg\\:max-w-\\[250px\\].xl\\:min-w-\\[325px\\].xl\\
                "div > div > div.group.flex.flex-col.md\\:flex-row-reverse.lg\\:flex-col.flex-wrap.md\\:items-cent
                "justify-between.gap-y-1.gap-x-2 > div:nth-child(2) > a > span"
            ))
        )
        # Берём родительский элемент <a>
        parent_anchor = website_span.find_element(By.XPATH, "..")
        return parent_anchor.get_attribute("href")

    except TimeoutException:
        print(f"[WARNING] На странице {company_url} не нашли ссылку на сайт (Timeout).")
        return None
    except NoSuchElementException:
        print(f"[WARNING] На странице {company_url} не нашли ссылку на сайт (NoSuchElement).")
        return None


def click_next_page(driver, max_retries=3):
    """
    Кликает на кнопку «Další» для перехода на следующую страницу результатов.
    Возвращает True, если переход был успешен, иначе False.
    """
    for attempt in range(max_retries):
        try:
            next_button = driver.find_element(
                By.CSS_SELECTOR,
                "#__nuxt > div > div.z-0.pb-3 > div.container.content-wrapper > "
                "div.w-full.\\[grid-area\\:results\\] > "
                "div.flex.flex-wrap.gap-y-2.items-center.justify-center.py-3.md\\:py-6 > a.button.next > img"
            )
            next_button.click()
            print("[INFO] Кликнули на кнопку 'Další' (попытка:", attempt + 1, ").")
            time.sleep(3)
            return True
        except (NoSuchElementException, StaleElementReferenceException, TimeoutException):
            # Небольшая пауза и ещё попытка
            print("[WARNING] Не удалось кликнуть 'Další'. Пауза и повтор.")
            time.sleep(2)
    # Если дошли сюда, значит не смогли кликнуть
    return False


def scrape_europages(query, country, num_results=5, start_page=1):
    """
    1) Открывает Europages с query в URL,
    2) Принимает cookies,
    3) Вводит страну (country), жмёт ENTER, затем 'Použít',
    4) Если надо – листаем «далее» (start_page – 1) раз, чтобы попасть на нужную страницу,
    5) Собираем ссылки (до num_results компаний), переходя по страницам результата.

    Возвращает кортеж (list результатов, int количество_собранных).
    """
    service = Service(CHROMEDRIVER_PATH)

    # Настраиваем ChromeOptions (например, чтобы не открывать окно на весь экран)
    chrome_options = webdriver.ChromeOptions()
    # Можно включить headless, если нужно работать без графического интерфейса:
    # chrome_options.add_argument("--headless")

    driver = webdriver.Chrome(service=service, options=chrome_options)
```

```
    driver.implicitly_wait(10)    # Неявное ожидание (на все find_element* методы)
    driver.set_page_load_timeout(60)  # Ждём до 60 секунд загрузку страниц

    results = []
    companies_collected = 0

    try:
        # 1. Открываем страницу с уже «вшитым» в URL поиском
        start_url = f"https://www.europages.cz/cs/search?q={query}"
        driver.get(start_url)
        print(f"[INFO] Открыли страницу Europages с запросом: {query}")

        # 2. Принимаем cookies, если всплывают
        accept_cookies(driver)
        time.sleep(2)

        # 3. Применяем фильтр по стране
        select_country_location(driver, country)

        # 4. Листаем до нужной start_page, если требуется
        #    (Если start_page=1, то этот цикл просто пропустится)
        for page_idx in range(1, start_page):
            print(f"[INFO] Пропускаем страницу №{page_idx}, кликаем 'Další'...")
            clicked = click_next_page(driver)
            if not clicked:
                print("[INFO] Не смогли дойти до start_page, 'Další' недоступна.")
                break
            time.sleep(1)

        # 5. Цикл по сбору ссылок
        while companies_collected < num_results:
            # Пытаемся получить контейнер результатов (с 1 попыткой перезагрузки при неудаче)
            for attempt_reload in range(2):  # Две попытки — 0 и 1
                try:
                    results_container = WebDriverWait(driver, 15).until(
                        EC.presence_of_element_located((By.CSS_SELECTOR, "div[data-test='search-results']"))
                    )
                    break  # Если получилось — выходим из цикла for
                except TimeoutException:
                    if attempt_reload == 0:
                        # Первая неудача — пытаемся перезагрузить
                        print("[WARNING] Контейнер результатов не прогрузился, пробуем перезагрузить страницу.")
                        driver.refresh()
                        time.sleep(5)
                    else:
                        # Вторая неудача — выходим из while
                        print("[ERROR] Контейнер результатов так и не прогрузился. Останавливаем сбор.")
                        return results, companies_collected
            else:
                # Если дошли до сюда, значит цикл for закончился без break
                break

            # Собираем карточки компаний
            company_links = results_container.find_elements(By.CSS_SELECTOR, "a.text-navy-100.font-display-400")
            total_this_page = len(company_links)
            print(f"[INFO] Найдено компаний на странице: {total_this_page}")

            # Перебираем компании на текущей странице
            for i in range(total_this_page):
                if companies_collected >= num_results:
                    break

                try:
                    # По возможности обновляем ссылку на контейнер и ссылки (борьба со StaleElement)
                    results_container = driver.find_element(By.CSS_SELECTOR, "div[data-test='search-results']")
                    company_links = results_container.find_elements(By.CSS_SELECTOR, "a.text-navy-100.font-display-

                    company_url = company_links[i].get_attribute("href")



                    results_container = driver.find_element(By.CSS_SELECTOR, "div[data-test='search-results']")
                    company_links = results_container.find_elements(By.CSS_SELECTOR, "a.text-navy-100.font-display-

                    company_url = company_links[i].get_attribute("href")
                    company_name = company_links[i].text.strip()

                    # Открываем компанию в новой вкладке
                    driver.execute_script("window.open(arguments[0]);", company_url)
                    driver.switch_to.window(driver.window_handles[-1])  # Переключаемся в новую вкладку

                    link = extract_website_link(driver, company_url)
                    if link:
                        results.append(f"{company_name}: {link}")
                    else:
                        results.append(f"{company_name}: Odkaz nenalezen")

                    companies_collected += 1

                    # Закрываем вкладку и возвращаемся
                    driver.close()
                    driver.switch_to.window(driver.window_handles[0])
                    time.sleep(1)

                except (TimeoutException, StaleElementReferenceException) as e:
                    print(f"[ERROR] Ошибка при обработке {companies_collected + 1}-й компании: {e}")
                    results.append(f"Společnost {companies_collected + 1}: Timeout/StaleElement chyba.")
                    companies_collected += 1
                    if len(driver.window_handles) > 1:
                        driver.close()
                        driver.switch_to.window(driver.window_handles[0])
                except Exception as e:
                    print("[ERROR] Неизвестная ошибка при обработке одной компании:", e)
                    results.append(f"Společnost {companies_collected + 1}: Neznámá chyba.")
                    companies_collected += 1
                    if len(driver.window_handles) > 1:
                        driver.close()
                        driver.switch_to.window(driver.window_handles[0])

            # Если ещё не набрали нужное количество, пытаемся перейти на след. страницу
            if companies_collected < num_results:
                clicked = click_next_page(driver)
                if not clicked:
                    print("[INFO] Кнопка 'Další' недоступна, заканчиваем сбор.")
                    break

        return results, companies_collected

    except Exception as e:
        print("[ERROR] Критическая ошибка в scrape_europages:", e)
        return results, companies_collected
    finally:
        driver.quit()


# ---- Пример локального запуска ----
if __name__ == "__main__":
    test_query = "elektronika"
    test_country = "Švýcarsko"
    test_num = 20
    # Начнём, например, со 2-й страницы
    test_start_page = 2

    results_data, collected_count = scrape_europages(
        query=test_query,
        country=test_country,
        num_results=test_num,
        start_page=test_start_page
    )



    print("\n--- VÝSLEDKY ---")
    for idx, item in enumerate(results_data, start=1):
        print(f"{idx}. {item}")

    # Пример: отправить на вебхук (если нужно)
    example_webhook_url = "https://hook.eu2.make.com/c8n5bo3ihwwe2kot16rvdsg8rwi5fje7"
    try:
        r = requests.post(
            example_webhook_url,
            json={"results": results_data, "companies_collected": collected_count},
            timeout=30
        )
        if r.status_code == 200:
            print("[INFO] Výsledky byly úspěšně odeslány na webhook.")
        else:
            print(f"[WARNING] Chyba při odesílání na webhook. Code: {r.status_code}")
    except Exception as ex:
        print("[ERROR] Problém při volání webhooku:", ex)
```

Zdroj: Autor

Příloha C Search_Gmail.py

```
import sys
import re
import json
import requests
from collections import deque
from urllib.parse import urljoin, urlparse

from bs4 import BeautifulSoup
from playwright.sync_api import sync_playwright, TimeoutError as PlaywrightTimeout
from flask import Flask, request, jsonify
import logging
from urllib.robotparser import RobotFileParser
import certifi
import spacy
from langdetect import detect, DetectorFactory
import stanza
import phonenumbers

DetectorFactory.seed = 0

# --- Настройка логирования ---
logging.basicConfig(
    level=logging.INFO,
    format="%(asctime)s - %(levelname)s - %(message)s"
)
logger = logging.getLogger(__name__)

# --- Регулярные выражения ---
LOOSE_EMAIL_REGEX = re.compile(
    r"[a-zA-Z0-9._%+\-]+@[a-zA-Z0-9.\-]+\.[a-zA-Z]{2,}"
)
STRICT_EMAIL_REGEX = re.compile(
    r"^[a-zA-Z0-9._%+\-]+@[a-zA-Z0-9.\-]+\.[a-zA-Z]{2,}$"
)

# Файлы, которые игнорируем (бинарные, медиа и т.п.)
BAD_EXTENSIONS = {
    ".jpg", ".jpeg", ".png", ".gif", ".pdf", ".svg", ".webp", ".bmp",
    ".doc", ".docx", ".xls", ".xlsx", ".ppt", ".pptx", ".zip", ".rar"
}

# --- Ключевые слова для приоритезации «контактных» страниц ---
PRIORITY_KEYWORDS = [
    "contact", "about", "team", "impressum", "kontakt", "kontakty", "contact-us",
    "contato", "contatti", "kontaktieren", "kontaktni", "contacto", "contactez",
    "联系", "контакт", "about-us", "o-nas", "sobre-nosotros", "über-uns",
    "chi-siamo", "à-propos", "om-oss", "locations", "offices", "branches",
    "filialen", "lokace", "standorte", "help", "support", "faq",
    "customer-service", "servis", "kundendienst", "departments", "sales",
    "marketing", "hr", "service", "logistics", "legal", "terms",
    "privacy-policy", "agb", "datenschutz", "feedback", "opinion", "review",
    "мнения", "отзывы", "get-in-touch", "form", "callback", "kontakt-formular",
    "kontakt.html", "contact.html", "about.html", "impressum.html", "gdpr"
]

# --- Синонимы для отделов (DEPARTAMENT) ---
DEPARTMENT_SYNONYMS = {
    "Marketing": [
        "marketing", "advertising", "advertise", "promotion", "promotions",
        "маркетинг", "реклама", "werbung", "publicité", "publicidad"
    ],
    "Sales": [
        "sales", "sale", "продажи", "verkauf", "venta", "ventas"
    ],
    "Support": [
        "support", "helpdesk", "help desk", "service desk",
        "поддержка", "служба поддержки", "обслуживание",
        "kundenservice". "kundendienst". "asistencia". "soutien"


        "kundenservice", "kundendienst", "asistencia", "soutien"
    ],
    "HR": [
        "hr", "human resources", "recruitment", "recruiting",
        "кадры", "отдел персонала", "отдел кадров",
        "personalabteilung", "ressources humaines", "recursos humanos",
        "personální"
    ],
    "Logistics": [
        "logistic", "logistics", "shipping", "delivery", "transport",
        "логистика", "доставка", "транспорт",
        "logistik", "livraison", "transporte"
    ],
    "Operations / Dispatch": [
        "provoz", "dispečink", "dispatch", "operational", "operations"
    ],
    "Information": [
        "info", "information", "allgemeine info", "общая информация",
        "информация", "información", "informationen", "informations"
    ]
}

# --- Синонимы для должностей (POZICE) ---
POSITION_SYNONYMS = {
    "Manager": [
        "manager", "mgr", "menedžer", "gestionnaire", "gerente",
        "managerin", "manager"
    ],
    "Director": [
        "director", "direktor", "директор", "chief", "head",
        "chef", "chief officer", "directeur", "ředitel",
        "jednatel"
    ],
    "Specialist": [
        "specialist", "специалист", "spezialist", "спеціаліст",
        "especialista", "spécialiste"
    ],
    "Assistant": [
        "assistant", "ассистент", "помощник", "assistent",
        "ayudante", "assistente"
    ],
    "Intern": [
        "intern", "trainee", "стажёр", "praktikant", "internship",
        "pasante", "stagiaire"
    ],
    "Coordinator": [
        "coordinator", "координатор", "koordinator", "coordinador"
    ],
    "Supervisor": [
        "supervisor", "супервайзер", "aufsicht", "superviseur"
    ],
    "Executive": [
        "executive", "исполнитель", "leitender", "ejecutivo", "cadre"
    ],
    "Lead": [
        "lead", "leader", "лидер", "главный", "líder",
        "leiter", "chef"
    ],
    "Administrator": [
        "administrator", "администратор", "admin", "administrateur",
        "administrador"
    ],
    "Consultant": [
        "consultant", "консультант", "berater", "consultora",
        "consulting"
    ],
    "Analyst": [
        "analyst". "аналитик". "analista". "analyste"
```

```
    "Analyst": [
        "analyst", "аналитик", "analista", "analyste"
    ],
    "Designer": [
        "designer", "дизайнер", "gestalter", "diseñador"
    ],
    "Developer": [
        "developer", "разработчик", "programmer", "programmiст",
        "software engineer", "инженер", "développeur", "programador"
    ],
    "Engineer": [
        "engineer", "инженер", "ingenieur", "ingénieur", "ingeniero"
    ],
    "Representative": [
        "representative", "представитель", "vertreter", "representante"
    ],
    "Strategist": [
        "strategist", "стратег", "stratege", "estratega"
    ],
    "Operator": [
        "operator", "оператор", "bediener", "operador"
    ],
    "Planner": [
        "planner", "планировщик", "planificador", "planer"
    ],
    "Technician": [
        "technician", "техник", "techniker", "técnico"
    ],
    "Recruiter": [
        "recruiter", "рекрутер", "recruteur", "reclutador"
    ],
    "Trainer": [
        "trainer", "тренер", "обучение", "инструктор", "ausbilder",
        "formateur", "entrenador"
    ],
    "Receptionist": [
        "receptionist", "ресепшионист", "recepcionista", "réceptionniste"
    ],
    "Accountant": [
        "accountant", "бухгалтер", "financier", "финансист",
        "buchhalter", "comptable", "contador"
    ],
    "Operations Manager": [
        "vedoucí provozu"
    ],
    "Branch Manager": [
        "vedoucí pobočky"
    ]
}

app = Flask(__name__)

# --- Кэш для robots.txt ---
robots_cache = {}

# --- Максимальное допустимое значение для max_pages ---
MAX_PAGES_LIMIT = 15

# --- Загрузка моделей spaCy ---
spacy_models = {}

def load_spacy_model(lang_code: str):
    """
    Пытаемся загрузить модель spaCy для заданного языка.
    Если модель отсутствует, ставим None.
    """
    global spacy_models
    trv:



    try:
        if lang_code == "en":
            spacy_models[lang_code] = spacy.load("en_core_web_sm")
        elif lang_code == "de":
            spacy_models[lang_code] = spacy.load("de_core_news_sm")
        elif lang_code == "ru":
            spacy_models[lang_code] = spacy.load("ru_core_news_sm")
        elif lang_code == "fr":
            spacy_models[lang_code] = spacy.load("fr_core_news_sm")
        elif lang_code == "es":
            spacy_models[lang_code] = spacy.load("es_core_news_sm")
        elif lang_code == "it":
            spacy_models[lang_code] = spacy.load("it_core_news_sm")
        elif lang_code == "pt":
            spacy_models[lang_code] = spacy.load("pt_core_news_sm")
        elif lang_code == "cs":
            # Официальной чешской модели нет
            spacy_models[lang_code] = None
        else:
            spacy_models[lang_code] = None
    except Exception as e:
        logger.warning(f"Не удалось загрузить модель spaCy для {lang_code}: {e}")
        spacy_models[lang_code] = None

# Поддерживаемые языки (при желании расширить)
supported_languages = ["en", "de", "ru", "fr", "es", "it", "pt", "cs"]
for lang in supported_languages:
    load_spacy_model(lang)


def is_valid_email_candidate(candidate: str) -> bool:
    candidate_lower = candidate.lower().strip()
    if not STRICT_EMAIL_REGEX.match(candidate_lower):
        return False
    for ext in BAD_EXTENSIONS:
        if candidate_lower.endswith(ext):
            return False
    return True


def extract_emails_from_html(html: str) -> list:
    """
    Ищем email + небольшой фрагмент (snippet) вокруг.
    Возвращаем [(email, snippet), ...].
    """
    soup = BeautifulSoup(html, "lxml")
    results = set()

    # mailto-ссылки
    for a_tag in soup.find_all("a", href=True):
        href = a_tag["href"]
        if href.startswith("mailto:"):
            email = href[7:].split("?")[0]
            if is_valid_email_candidate(email):
                parent = a_tag.find_parent(['div','li','p','span','td'])
                snippet = parent.get_text(separator=' ', strip=True) if parent else ""
                results.add((email, snippet))

    # Сырые email-адреса
    for match in LOOSE_EMAIL_REGEX.finditer(html):
        raw_email = match.group(0).strip()
        if is_valid_email_candidate(raw_email):
            start_idx = max(0, match.start() - 100)
            end_idx = min(len(html), match.end() + 100)
            snippet_raw = html[start_idx:end_idx]
            snippet_text = BeautifulSoup(snippet_raw, "lxml").get_text(separator=' ', strip=True)
            results.add((raw_email, snippet_text))
```

```
    return list(results)


def extract_phone_numbers(text: str) -> list:
    """
    Используем phonenumbers для извлечения номеров.
    Возвращаем уникальные номера в международном формате.
    """
    unique_phones = set()
    for match in phonenumbers.PhoneNumberMatcher(text, "ZZ"):
        phone_obj = match.number
        if phonenumbers.is_possible_number(phone_obj) and phonenumbers.is_valid_number(phone_obj):
            phone_str = phonenumbers.format_number(
                phone_obj,
                phonenumbers.PhoneNumberFormat.INTERNATIONAL
            )
            unique_phones.add(phone_str)
    return sorted(unique_phones)


def parse_name_from_email(email: str) -> str:
    """
    Пытаемся «выделить» имя из локальной части email (до @).
    Если не вышло — возвращаем "".
    """
    local_part = email.split("@", 1)[0]
    cleaned = re.sub(
        r"[^a-zA-Za-яА-ЯёЁ.\-]",
        " ",
        local_part,
        flags=re.IGNORECASE
    )
    parts = re.split(r"[.\-]+", cleaned)
    parts = [p.strip() for p in parts if len(p.strip()) > 1]
    if not parts:
        return ""
    name_parts = [p.capitalize() for p in parts]
    full_name = " ".join(name_parts)
    if len(full_name) < 3:
        return ""
    return full_name


def guess_department(email: str, snippet: str) -> str:
    text_lower = (email + " " + snippet).lower()
    for dept_name, keywords in DEPARTMENT_SYNONYMS.items():
        for kw in keywords:
            if kw in text_lower:
                return dept_name
    return ""


def guess_position(email: str, snippet: str, lang_code: str) -> str:
    """
    Пытаемся вычислить должность (position) через spaCy (NER) + словари.
    """
    text_lower = (email + " " + snippet).lower()
    # 1) Попытка spaCy NER
    nlp = spacy_models.get(lang_code)
    if nlp:
        try:
            doc = nlp(snippet)
            for ent in doc.ents:
                ent_lower = ent.text.lower()
                for pos_name, pos_keywords in POSITION_SYNONYMS.items():
                    for kw in pos_keywords:
                        if kw in ent_lower:


                for pos_name, pos_keywords in POSITION_SYNONYMS.items():
                    for kw in pos_keywords:
                        if kw in ent_lower:
                            return pos_name
        except Exception as e:
            logger.warning(f"spaCy ошибка ({lang_code}): {e}")

    # 2) Фолбэк: словари
    for pos_name, pos_keywords in POSITION_SYNONYMS.items():
        for kw in pos_keywords:
            if kw in text_lower:
                return pos_name

    return ""


def is_allowed_by_robots(url: str) -> bool:
    """
    Проверяем robots.txt. Если нет или не вышло скачать — True.
    """
    parsed_url = urlparse(url)
    robots_url = urljoin(
        f"{parsed_url.scheme}://{parsed_url.netloc}",
        "/robots.txt"
    )
    if robots_url in robots_cache:
        rp = robots_cache[robots_url]
    else:
        rp = RobotFileParser()
        try:
            response = requests.get(
                robots_url,
                headers={'User-Agent': 'Mozilla/5.0'},
                timeout=10,
                verify=certifi.where()
            )
            if response.status_code == 200:
                rp.parse(response.text.splitlines())
                robots_cache[robots_url] = rp
            else:
                return True
        except Exception:
            return True

    return rp.can_fetch("*", url)


def prioritize_contact_pages(links: list) -> list:
    """
    Возвращает список ссылок, где «контактные» идут первыми.
    """
    prioritized = []
    non_prioritized = []
    for link in set(links):
        link_lower = link.lower()
        score = sum(kw in link_lower for kw in PRIORITY_KEYWORDS)
        if score > 0:
            prioritized.append((link, score))
        else:
            prioritized.append((link, 0))

    # Сортируем по убыванию score
    sorted_links = sorted(prioritized, key=lambda x: x[1], reverse=True)
    return [link for (link, score) in sorted_links]


def parse_page_and_popups(page) -> list:
    """
```

```
    """
    Возвращает список (email, snippet).
    Параллельно пытаемся «открыть» попапы, если есть.
    """
    results = set()

    normal_html = page.content()
    normal_emails = extract_emails_from_html(normal_html)
    results.update(normal_emails)

    try:
        popup_triggers = page.query_selector_all('[data-open-as="popup"]')
    except Exception as e:
        logger.error(f"Ошибка поиска попап-триггеров: {e}")
        popup_triggers = []

    for i, trigger in enumerate(popup_triggers):
        try:
            trigger.click()
            page.wait_for_selector("main.popup", timeout=5000)
            popup_html = page.content()
            popup_emails = extract_emails_from_html(popup_html)
            results.update(popup_emails)
            page.keyboard.press("Escape")
        except PlaywrightTimeout:
            logger.warning(f"Попап #{i+1} не загрузился")
        except Exception as e:
            logger.error(f"Ошибка в попапе #{i+1}: {e}")

    return list(results)


def crawl_page(url: str, page_content_callback) -> dict:
    """
    - Открывает страницу.
    - Ищет email + snippet, phone, department, position, name.
    - Возвращает словарь формата:
      {
        "email1": {
          "name": "...",
          "department": "...",
          "position": "...",
          "phones": [...]
        },
        ...
      }
    """
    people_map = {}
    page = page_content_callback() if page_content_callback else None
    if not page:
        return people_map

    try:
        email_infos = parse_page_and_popups(page)
    except Exception as e:
        logger.error(f"Ошибка попапов на странице {url}: {e}")
        email_infos = []

    page.close()

    for (email, snippet) in email_infos:
        # Определяем язык
        try:
            lang = detect(snippet)
            # Если нет модели, берём "en"
            if lang not in spacy_models:
                lang = "en"
        except:


        # Телефоны, имя, отдел, позиция
        phones = extract_phone_numbers(snippet)
        name = parse_name_from_email(email)
        dept = guess_department(email, snippet)
        pos = guess_position(email, snippet, lang)

        if email not in people_map:
            people_map[email] = {
                "name": name,
                "department": dept,
                "position": pos,
                "phones": phones
            }
        else:
            # Дополняем, если уже есть
            if name and not people_map[email]["name"]:
                people_map[email]["name"] = name
            if dept and not people_map[email]["department"]:
                people_map[email]["department"] = dept
            if pos and not people_map[email]["position"]:
                people_map[email]["position"] = pos

            merged_phones = set(people_map[email]["phones"]) | set(phones)
            people_map[email]["phones"] = sorted(merged_phones)

    return people_map


def deep_crawl_people_with_departments(start_url: str, max_pages=15):
    """
    Глубокий обход (BFS) до max_pages.
    Возвращает результат в формате параллельных списков:
    {
      "email": {"list": [...]},
      "name": {"list": [...]},
      "telefon": {"list": [...]},
      "departament": {"list": [...]},
      "pozice": {"list": [...]}
    }
    """
    visited = set()
    queue = deque([start_url])
    global_map = {}

    with sync_playwright() as p:
        browser = p.chromium.launch(headless=True)
        context = browser.new_context()

        while queue and len(visited) < max_pages:
            url = queue.popleft()
            if url in visited:
                continue
            visited.add(url)

            # Проверяем robots.txt
            if not is_allowed_by_robots(url):
                logger.info(f"Отклонено robots.txt: {url}")
                continue

            parsed = urlparse(url)
            if parsed.scheme not in ("http", "https"):
                continue

            def page_content_callback():
                try:
                    page = context.new_page()
                    page.goto(url, timeout=15000, wait_until="domcontentloaded")
                    page.wait_for_timeout(2000)
```

```
                    page.wait_for_timeout(2000)
                    return page
                except Exception as e:
                    logger.error(f"Ошибка при загрузке {url}: {e}")
                    return None

            # Собираем контакты с текущей страницы
            page_result = crawl_page(url, page_content_callback)

            # Сохраняем в global_map
            for email, data in page_result.items():
                if email not in global_map:
                    global_map[email] = data
                else:
                    if data["name"] and not global_map[email]["name"]:
                        global_map[email]["name"] = data["name"]
                    if data["department"] and not global_map[email]["department"]:
                        global_map[email]["department"] = data["department"]
                    if data["position"] and not global_map[email]["position"]:
                        global_map[email]["position"] = data["position"]
                    merged_phones = set(global_map[email]["phones"]) | set(data["phones"])
                    global_map[email]["phones"] = sorted(merged_phones)

            # Собираем новые ссылки
            html = ""
            try:
                tmp_page = context.new_page()
                tmp_page.goto(url, timeout=15000, wait_until="domcontentloaded")
                tmp_page.wait_for_timeout(2000)
                html = tmp_page.content()
                tmp_page.close()
            except Exception as e:
                logger.error(f"Ошибка при извлечении ссылок {url}: {e}")

            if html:
                soup = BeautifulSoup(html, "lxml")
                found_links = [a.get("href") for a in soup.find_all("a", href=True)]
                valid_links = []
                for lk in found_links:
                    if lk and urlparse(lk).scheme in ("http", "https"):
                        # Пропускаем бинарные файлы
                        if not any(lk.lower().endswith(ext) for ext in BAD_EXTENSIONS):
                            valid_links.append(lk)

                prio_links = prioritize_contact_pages(valid_links)
                abs_links = [urljoin(url, lk) for lk in prio_links]
                for link_candidate in abs_links:
                    if link_candidate not in visited and link_candidate not in queue:
                        queue.append(link_candidate)

        browser.close()

    # Преобразуем global_map -> параллельные списки
    people_temp = []
    for email, data in global_map.items():
        people_temp.append({
            "email": email,
            "name": data.get("name", ""),
            "phones": data.get("phones", []),
            "department": data.get("department", ""),
            "position": data.get("position", "")
        })

    # Сортируем по email
    people_temp.sort(key=lambda x: x["email"])

    email_list = []
    name_list = []


    phone_list = []
    department_list = []
    position_list = []

    for person in people_temp:
        email_list.append(person["email"])
        name_list.append(person["name"] if person["name"] else "")
        phones_str = ", ".join(person["phones"]) if person["phones"] else ""
        phone_list.append(phones_str)
        department_list.append(person["department"] if person["department"] else "")
        position_list.append(person["position"] if person["position"] else "")

    return {
        "email": {"list": email_list},
        "name": {"list": name_list},
        "telefon": {"list": phone_list},
        "departament": {"list": department_list},
        "pozice": {"list": position_list}
    }


@app.route('/process', methods=['POST'])
def process():
    data = request.json
    logger.info(f"Получены данные: {data}")
    link = data.get("link")
    max_pages = data.get("max_pages", 15)

    if not isinstance(max_pages, int) or not (1 <= max_pages <= MAX_PAGES_LIMIT):
        return jsonify({"error": f"max_pages должен быть в диапазоне [1..{MAX_PAGES_LIMIT}]."}), 400

    if not link or not urlparse(link).scheme:
        return jsonify({"error": "Некорректный или пустой URL"}), 400

    try:
        final_result = deep_crawl_people_with_departments(link, max_pages=max_pages)
        return jsonify(final_result), 200
    except Exception as e:
        logger.error(f"Ошибка при обработке {link}: {e}")
        return jsonify({"error": str(e)}), 500


if __name__ == "__main__":
    # Можно указать порт как аргумент, например: python Search_Gmail.py 5000
    port = 5000
    if len(sys.argv) > 1:
        try:
            port = int(sys.argv[1])
        except ValueError:
            pass

    app.run(host='0.0.0.0', port=port)
```

Zdroj: Autor

Příloha D linkedin_sender.py

```python
import os
import json
import time
import random
import logging
import schedule
from pathlib import Path
from typing import List
from datetime import datetime
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.action_chains import ActionChains
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.support.ui import WebDriverWait, Select
from selenium.webdriver.support import expected_conditions as EC
from selenium.common.exceptions import (
    NoSuchElementException,
    TimeoutException,
    StaleElementReferenceException,
    ElementClickInterceptedException
)
from webdriver_manager.chrome import ChromeDriverManager


# Настройка логирования
logging.basicConfig(
    level=logging.INFO,
    format='[%(asctime)s] %(levelname)s: %(message)s',
    datefmt='%Y-%m-%d %H:%M:%S'
)

PROGRESS_FILE = "progress.json"
PROFILES_FILE = "profiles.txt"

DAILY_LIMIT = 70    # Максимум сообщений в день
WORK_START = 6      # Начало рабочего окна (час)
WORK_END = 17       # Конец рабочего окна (час)

class InvalidProfileException(Exception):
    """Исключение, указывающее, что профиль недопустим."""
    pass

def extract_company_name(profile_url: str) -> str:
    """
    Извлекает название компании из URL, например:
    https://www.linkedin.com/company/hatz-diesel/about -> hatz-diesel
    https://www.linkedin.com/company/hatz-diesel -> hatz-diesel

    Если не удаётся корректно извлечь, возвращаем 'your company' (по умолчанию).
    """
    url = profile_url.strip().lower().rstrip("/")
    parts = url.split("/")
    try:
        company_index = parts.index("company")
        if company_index + 1 < len(parts):
            candidate = parts[company_index + 1]
            if candidate in ("about", "life", "jobs"):
                if company_index + 2 < len(parts):
                    candidate = parts[company_index + 2]
                else:
                    candidate = None
            if candidate and candidate not in ("about", "life", "jobs"):
                return candidate
    except ValueError:
        pass
    return "your company"


class LinkedInCompanyBot:
    """
    Бот для автоматизации рассылки сообщений на страницы компаний LinkedIn.
    """

    LINKEDIN_LOGIN_URL = "https://www.linkedin.com/login"

    def __init__(self, email: str, password: str, daily_limit: int = 70) -> None:
        self.email = email
        self.password = password
        self.daily_limit = daily_limit

        self.driver = self.create_browser()

        # Счётчики статистики
        self.success_count = 0
        self.error_count = 0
        self.invalid_count = 0

        # Сообщения для компаний (20 вариантов)
        self.company_messages = self.prepare_company_messages()

    def create_browser(self) -> webdriver.Chrome:
        """
        Создает и настраивает браузер Chrome с нужными опциями.
        """
        options = Options()
        options.add_argument("--disable-blink-features=AutomationControlled")
        options.add_argument("start-maximized")
        options.add_argument("--disable-notifications")
        options.add_argument(
            "user-agent=Mozilla/5.0 (Windows NT 10.0; Win64; x64) "
            "AppleWebKit/537.36 (KHTML, like Gecko) "
            "Chrome/110.0.5481.177 Safari/537.36"
        )
        driver = webdriver.Chrome(
            service=Service(ChromeDriverManager().install()),
            options=options
        )
        return driver

    def prepare_company_messages(self) -> List[str]:
        """
        Шаблоны сообщений для компаний с использованием {company}.
        """
        return [
            "Good day! This is Nikita, Senior Sales Manager at LOG-IN CZ s.r.o. We specialize in secure logistics solutions tailored for high-value cargo. I'd appreciate an opportunity to explore pote
            "Hello, Nikita here from LOG-IN CZ s.r.o. We offer professional and secure logistics solutions specifically designed for companies like {company}. Could we arrange a time to discuss this f
            "Greetings! Nikita from LOG-IN CZ s.r.o. reaching out. Our company provides specialized transport services for valuable and sensitive goods. I believe we can offer effective logistics solu
            "Good afternoon! Nikita, Senior Sales Manager at LOG-IN CZ s.r.o., here. We deliver highly secure and efficient logistics tailored to your unique business needs. Would {company} be open to
            "Hi there, this is Nikita from LOG-IN CZ s.r.o. Our expertise is secure transportation of high-value goods across Europe. I'm keen to discuss how we could support {company} in your logisti
            "Hello, Nikita here, representing LOG-IN CZ s.r.o. We specialize in logistics for valuable and bulky cargo, ensuring security and reliability at every stage. Could we schedule a conversati
            "Good day! Nikita from LOG-IN CZ s.r.o. contacting you. Our tailored logistics solutions have successfully supported businesses similar to {company}. Would you be interested in exploring a
            "Hello, this is Nikita from LOG-IN CZ s.r.o. We pride ourselves on delivering secure logistics services for high-value products. I think our solutions could greatly benefit {company}. Woul
            "Greetings! Nikita here from LOG-IN CZ s.r.o. Our dedicated logistics services focus on the safe transportation of valuable goods. I'd like to propose a discussion about potential cooperat
            "Good afternoon! This is Nikita, Senior Sales Manager at LOG-IN CZ s.r.o. We specialize in reliable logistics for high-value and sensitive cargo, and I would appreciate exploring how our s
            "Hello, Nikita from LOG-IN CZ s.r.o. We have proven logistics solutions designed specifically for businesses handling high-value cargo like {company}. Would you be interested in a brief co
            "Good day! Nikita, Senior Sales Manager at LOG-IN CZ s.r.o., here. We offer comprehensive logistics services tailored for valuable goods. Could we briefly discuss how our expertise could b
            "Hi, Nikita from LOG-IN CZ s.r.o. We specialize in secure and efficient transport for businesses like {company}. Could we explore a potential partnership together? Visit https://www.login-
            "Good afternoon, Nikita here from LOG-IN CZ s.r.o. Our logistics expertise ensures the secure handling and transportation of high-value products. I would be glad to discuss cooperation opp
            "Hello, this is Nikita from LOG-IN CZ s.r.o. We excel at delivering secure logistics tailored to specific business needs. I'd like to discuss how we might support {company}. Interested in
            "Greetings! Nikita from LOG-IN CZ s.r.o. here. We are experts in the logistics of valuable cargo, and I'd welcome the chance to explore potential collaboration with {company}. Please visit
            "Good day, Nikita speaking from LOG-IN CZ s.r.o. Our logistics services focus specifically on secure transport for high-value goods. Could we schedule a short discussion about working toge
            "Hello! Nikita, Senior Manager at LOG-IN CZ s.r.o. here. We're dedicated to providing secure and professional logistics services. I'd appreciate an opportunity to discuss how we could help
            "Hi, Nikita from LOG-IN CZ s.r.o. reaching out. We provide reliable logistics solutions for companies dealing with valuable cargo, like {company}. Would you be interested in discussing thi
            "Good day! Nikita from LOG-IN CZ s.r.o. here. Our specialized logistics solutions are designed to secure high-value shipments. Could we arrange a discussion about potential collaboration w
        ]
```

```
        "Good day! Nikita from LOG-IN CZ s.r.o. here. Our specialized logistics solutions are c
    ]

def random_sleep(self, min_sec: float = 3, max_sec: float = 7) -> None:
    """
    Выполняет случайную задержку между действиями.
    """
    delay = random.uniform(min_sec, max_sec)
    logging.info(f"Имитируем задержку: ~{delay:.2f} сек...")
    time.sleep(delay)

def simulate_human_behavior(self, min_scrolls: int = 1, max_scrolls: int = 3) -> None:
    """
    Имитация "человеческого" поведения: случайный скролл страницы, небольшие паузы и т.д.
    """
    scroll_count = random.randint(min_scrolls, max_scrolls)
    logging.info(f"Имитируем поведение: скроллим страницу {scroll_count} раз...")
    for i in range(scroll_count):
        scroll_distance = random.randint(200, 800)
        self.driver.execute_script(f"window.scrollBy(0, {scroll_distance});")
        self.random_sleep(2, 5)
        # Возможен скролл назад
        if random.choice([True, False]):
            self.driver.execute_script(f"window.scrollBy(0, {-scroll_distance // 2});")
            self.random_sleep(1, 3)

def login(self) -> None:
    """
    Авторизуется на LinkedIn.
    """
    logging.info("Попытка входа в систему...")
    self.driver.get(self.LINKEDIN_LOGIN_URL)
    self.random_sleep(4, 8)
    try:
        username_input = self.driver.find_element(By.ID, "username")
        password_input = self.driver.find_element(By.ID, "password")
        username_input.send_keys(self.email)
        self.random_sleep(1, 2)
        password_input.send_keys(self.password)
        password_input.send_keys(Keys.RETURN)

        WebDriverWait(self.driver, 15).until(
            EC.presence_of_element_located((By.ID, "global-nav-search"))
        )
        self.random_sleep(2, 4)
        logging.info(f"Успешный вход: {self.email}")
    except Exception:
        logging.error("Ошибка при входе в систему.", exc_info=True)
        self.error_count += 1

def scroll_and_click(self, element) -> None:
    """
    Прокручивает страницу к элементу и кликает по нему через JavaScript.
    При неудаче пробует ActionChains.
    """
    try:
        self.driver.execute_script("arguments[0].scrollIntoView({block: 'center'});", element)
        self.random_sleep(1, 2)
        self.driver.execute_script("arguments[0].click();", element)
        self.random_sleep(1, 2)
    except Exception:
        logging.error("Ошибка при JS-клике.", exc_info=True)
        try:
            actions = ActionChains(self.driver)
            actions.move_to_element(element).pause(1).click(element).perform()
            self.random_sleep(1, 2)
        except Exception:
            logging.error("Ошибка при ActionChains-клике.", exc_info=True)



def close_chat_overlay(self) -> None:
    """
    Сворачивает окно чата, если оно мешает.
    """
    try:
        chat_overlay = self.driver.find_element(By.ID, "msg-overlay")
        minimize_btn = chat_overlay.find_element(By.XPATH, ".//button[contains(@aria-label,'Minimize')]")
        self.scroll_and_click(minimize_btn)
        logging.info("Чат свёрнут (msg-overlay).")
    except NoSuchElementException:
        pass

def close_other_popups(self) -> None:
    """
    Закрывает всплывающие подсказки и окна.
    """
    try:
        dismiss_buttons = self.driver.find_elements(By.XPATH, "//button[contains(@aria-label, 'Dismiss')]")
        for btn in dismiss_buttons:
            self.scroll_and_click(btn)
            logging.info("Закрыта всплывающая подсказка (Dismiss).")
    except Exception:
        logging.error("Ошибка при закрытии всплывающих окон.", exc_info=True)

def cancel_discard_popup(self) -> None:
    """
    Отменяет появление окна с предложением 'Discard message?'.
    """
    try:
        self.driver.find_element(By.XPATH, "//div[contains(text(), 'Discard message')]")
        cancel_button = self.driver.find_element(By.XPATH, "//button[contains(text(), 'Cancel')]")
        logging.warning("Появилось окно 'Discard message'. Нажимаем 'Cancel'.")
        self.scroll_and_click(cancel_button)
        self.random_sleep(1, 2)
    except NoSuchElementException:
        pass

def go_to_profile(self, profile_url: str) -> bool:
    """
    Переходит на страницу (company) профиля.
    Проверяет, что URL не содержит 'unavailable' или '/about/'.
    """
    try:
        self.driver.get(profile_url)
        self.random_sleep(5, 9)
        self.simulate_human_behavior()

        final_url = self.driver.current_url.lower()
        if "unavailable" in final_url or "/about/" in final_url:
            logging.info(f"Страница недоступна или /about/: {profile_url}")
            self.invalid_count += 1
            raise InvalidProfileException("Неправильная ссылка (unavailable/about).")

        # Ожидаем, что загрузится некий заголовок
        WebDriverWait(self.driver, 15).until(
            EC.presence_of_element_located((By.XPATH, "//h1"))
        )
        logging.info(f"Открыта страница: {profile_url}")
        return True

    except InvalidProfileException:
        raise
    except Exception as e:
        logging.warning(f"Ошибка при переходе на профиль {profile_url}: {e}", exc_info=True)
        self.error_count += 1
        return False
```

```python
    def find_send_message_button(self):
        """
        Ищет кнопку 'Message' на странице компании.
        """
        try:
            message_button = WebDriverWait(self.driver, 10).until(
                EC.element_to_be_clickable(
                    (By.XPATH, "//button[contains(@aria-label, 'Message')] | //button[contains(text(), 'Message')]")
                )
            )
            return message_button
        except:
            return None

    def click_send_message(self, attempt: int):
        """
        Ищем и кликаем на кнопку 'Send message' (в диалоговой форме).
        """
        try:
            # Кнопка может быть в виде span внутри button
            span = self.driver.find_element(By.XPATH, "//span[@class='artdeco-button__text' and text()='Send message']")
            button = span.find_element(By.XPATH, "./ancestor::button")
            if button.get_attribute("disabled") == "true":
                logging.warning(f"Попытка {attempt}: 'Send message' отключена.")
                return False

            self.scroll_and_click(button)
            self.random_sleep(2, 3)
            self.cancel_discard_popup()
            return True
        except (NoSuchElementException, StaleElementReferenceException, ElementClickInterceptedException):
            logging.warning(f"Попытка {attempt}: не удалось кликнуть 'Send message'.", exc_info=True)
            self.cancel_discard_popup()
            return False

    def message_company(self, profile_url: str) -> bool:
        """
        Отправляет сообщение странице компании.
        """
        # 1. Проверяем, что это именно company-ссылка
        if "/company/" not in profile_url.lower():
            logging.info(f"Ссылка не является компанией: {profile_url}. Пропускаем.")
            return False

        # 2. Переходим на профиль
        if not self.go_to_profile(profile_url):
            return False

        # 3. Закрываем лишние оверлеи
        self.close_chat_overlay()
        self.close_other_popups()

        # 4. Ищем кнопку "Message"
        msg_button = self.find_send_message_button()
        if not msg_button:
            logging.info(f"Кнопка 'Message' не найдена на {profile_url}. Пропускаем.")
            return False

        # 5. Пытаемся нажать на кнопку для открытия формы
        self.scroll_and_click(msg_button)
        self.random_sleep(2, 4)

        # 6. Часто появляется окно с Select 'Other'
        #    Если не появится — обработаем исключение.
        try:
            WebDriverWait(self.driver, 5).until(
                EC.presence_of_element_located((By.ID, "msg-shared-modals-msg-page-modal-presenter-conversation-topic"))



            select_elem = self.driver.find_element(By.ID, "msg-shared-modals-msg-page-modal-presenter-conversation-topic")
            Select(select_elem).select_by_visible_text("Other")
            self.random_sleep(1, 2)

            text_area = WebDriverWait(self.driver, 5).until(
                EC.presence_of_element_located((By.ID, "org-message-page-modal-message"))
            )
            # Подбираем текст сообщения
            message_template = random.choice(self.company_messages)
            company_name = extract_company_name(profile_url)
            message = message_template.replace("{company}", company_name)

            # Имитация по-символьного ввода
            for ch in message:
                text_area.send_keys(ch)
                time.sleep(random.uniform(0.03, 0.10))

            self.random_sleep(1, 2)

            # 7. Пытаемся нажать кнопку "Send message" (3 попытки)
            for attempt in range(1, 4):
                if self.click_send_message(attempt):
                    logging.info(f"Сообщение отправлено на {profile_url}")
                    self.success_count += 1
                    return True
                else:
                    self.close_chat_overlay()
                    self.close_other_popups()
                    self.random_sleep(2, 4)

            logging.error(f"Не удалось нажать 'Send message' после 3 попыток: {profile_url}")
            return False

        except TimeoutException:
            # Если форма с Select'ом не появилась, бывает другой интерфейс. Можно расширить логику под разные сценарии.
            logging.info("Окно с Select('Other') не появилось, пропускаем профиль.")
            return False
        except Exception as e:
            logging.error(f"Ошибка при отправке сообщения компании {profile_url}: {e}", exc_info=True)
            self.error_count += 1
            return False

    def load_profiles(self) -> List[str]:
        """
        Загружает список URL-профилей из файла PROFILES_FILE.
        """
        p = Path(PROFILES_FILE)
        if not p.exists():
            logging.error(f"Файл {p} не найден!")
            return []
        try:
            with p.open("r", encoding="utf-8") as f:
                profiles = [line.strip() for line in f if line.strip()]
            logging.info(f"Загружено {len(profiles)} профилей из {p}.")
            return profiles
        except Exception:
            logging.error("Ошибка при загрузке профилей.", exc_info=True)
            self.error_count += 1
            return []

def load_progress() -> int:
    """
    Загружает индекс текущего профиля из JSON-файла progress.json.
    Если файл не существует или пуст, возвращает 0.
    """
    p = Path(PROGRESS_FILE)
    if not p.exists():
        return 0
```

```
    p = Path(PROGRESS_FILE)
    if not p.exists():
        return 0
    try:
        data = json.loads(p.read_text(encoding="utf-8"))
        return data.get("current_index", 0)
    except Exception:
        return 0

def save_progress(current_index: int) -> None:
    """
    Сохраняет индекс текущего профиля в JSON-файл progress.json.
    """
    data = {"current_index": current_index}
    Path(PROGRESS_FILE).write_text(json.dumps(data, ensure_ascii=False), encoding="utf-8")

def daily_run(bot: LinkedInCompanyBot) -> None:
    """
    Основная логика ежедневной работы.
    Отправляет сообщения компаниям до 17:00 или пока не достигнут DAILY_LIMIT.
    """
    logging.info("=== Начало ежедневной задачи (daily_run) ===")

    profiles = bot.load_profiles()
    if not profiles:
        logging.error("Нет профилей для обработки.")
        return

    current_index = load_progress()
    logging.info(f"Текущий индекс: {current_index}")

    sent_today = 0

    while True:
        now = datetime.now()
        if now.hour >= WORK_END:
            logging.info("Время >= 17:00. Завершаем отправку на сегодня.")
            break

        if sent_today >= DAILY_LIMIT:
            logging.info(f"Достигнут лимит {DAILY_LIMIT} сообщений за сегодня.")
            break

        if current_index >= len(profiles):
            logging.info("Список профилей закончился.")
            break

        profile_url = profiles[current_index]
        logging.info(f"Обрабатываем профиль #{current_index}: {profile_url}")

        try:
            success = bot.message_company(profile_url)
            if success:
                sent_today += 1
                logging.info("Сообщение успешно отправлено.")
                bot.random_sleep(5, 15)
            else:
                logging.info("Не удалось отправить сообщение, делаем краткую паузу.")
                bot.random_sleep(4, 10)

        except InvalidProfileException:
            logging.info(f"Профиль {profile_url} недопустим, переходим к следующему.")
            current_index += 1
            save_progress(current_index)
            continue

        current_index += 1
        save_progress(current_index)



        # Дополнительная пауза между профилями
        pause_minutes = random.randint(2, 5)
        logging.info(f"Пауза {pause_minutes} минут до следующего профиля...")
        time.sleep(pause_minutes * 60)

    logging.info("=== Статистика за день ===")
    logging.info(f"Отправлено сегодня: {sent_today}")
    logging.info(f"Успешных отправок (всего): {bot.success_count}")
    logging.info(f"Ошибок (всего): {bot.error_count}")
    logging.info(f"Неправильных ссылок (всего): {bot.invalid_count}")
    logging.info("=== Завершение daily_run ===")

def main() -> None:
    # Учётные данные (можно задать через переменные окружения)
    EMAIL = os.getenv("LINKEDIN_EMAIL", "crayfishpyatigorsk@gmail.com")
    PASSWORD = os.getenv("LINKEDIN_PASSWORD", "Cfifnikita2003")

    bot = LinkedInCompanyBot(EMAIL, PASSWORD, daily_limit=DAILY_LIMIT)
    bot.login()

    now = datetime.now()
    if WORK_START <= now.hour < WORK_END:
        logging.info("Текущее время в пределах рабочего интервала, запускаем daily_run.")
        daily_run(bot)

    # Планирование ежедневного запуска в 06:00
    schedule.every().day.at("06:00").do(daily_run, bot=bot)
    logging.info("Бот запущен и ожидает расписанного запуска (каждый день в 06:00).")

    try:
        while True:
            schedule.run_pending()
            time.sleep(30)
    except KeyboardInterrupt:
        logging.info("Остановка бота (Ctrl + C).")
    finally:
        bot.driver.quit()
        logging.info("Браузер закрыт. Завершение работы.")

if __name__ == "__main__":
    main()
```

Zdroj: Autor

Příloha E facebook_sender.py

```
import os
import json
import time
import random
import logging
import schedule
from pathlib import Path
from typing import List, Union
from datetime import datetime

import ssl
import certifi

# Функция, возвращающая корректный SSLContext с сертификатами из certifi
def use_certifi_context():
    return ssl.create_default_context(cafile=certifi.where())

# Переопределяем стандартный контекст, чтобы Python мог проверять сертификаты
ssl._create_default_https_context = use_certifi_context

# Используем undetected-chromedriver
import undetected_chromedriver as uc

from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver import ActionChains
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.common.exceptions import TimeoutException, NoSuchElementException

# ----------------------------
# Конфигурация логирования
# ----------------------------
logging.basicConfig(
    level=logging.INFO,
    format='[%(asctime)s] %(levelname)s: %(message)s',
    datefmt='%Y-%m-%d %H:%M:%S'
)

# ----------------------------
# Константы и настройки
# ----------------------------
PROGRESS_FILE_FB_1 = "fb_progress_1.json"
PROGRESS_FILE_FB_2 = "fb_progress_2.json"

# Файл со списком профилей (список URL)
PROFILES_FILE_FB = "fb_profiles.txt"

# Лимит сообщений в день — по 15 на каждый из двух аккаунтов
DAILY_LIMIT_1 = 15
DAILY_LIMIT_2 = 15

# Временные рамки для работы (06:00–17:00)
WORK_START = 6
WORK_END = 17

# Список User-Agent для случайного выбора — можно дополнять
USER_AGENTS = [
    "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 "
    "(KHTML, like Gecko) Chrome/110.0.5481.177 Safari/537.36",
    "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 "
    "(KHTML, like Gecko) Chrome/105.0.0.0 Safari/537.36",
    "Mozilla/5.0 (Macintosh; Intel Mac OS X 13_3) AppleWebKit/537.36 "
    "(KHTML, like Gecko) Chrome/111.0.0.0 Safari/537.36",
    "Mozilla/5.0 (Windows NT 10.0; rv:110.0) Gecko/20100101 Firefox/110.0",
    "Mozilla/5.0 (iPhone; CPU iPhone OS 15_5 like Mac OS X) "
    "AppleWebKit/605.1.15 (KHTML, like Gecko) Version/15.5 Mobile/15E148 Safari/604.1"
]



def extract_page_name(url: str) -> str:
    """
    Пытаемся извлечь название страницы из URL вида:
      https://www.facebook.com/SomePageName/
      https://facebook.com/SomePageName-12345
      https://facebook.com/profile.php?id=12345 (вернём 'profile.php?id=12345')
    Если не удаётся — возвращаем 'your page'.
    """
    url = url.strip().lower().rstrip('/')
    if 'facebook.com/' not in url:
        return "your page"

    parts = url.split('/')
    fb_indices = [i for i, part in enumerate(parts) if "facebook.com" in part]
    if not fb_indices:
        return "your page"
    fb_index = fb_indices[0]

    if fb_index + 1 < len(parts):
        candidate = parts[fb_index + 1]
        if candidate:
            return candidate
    return "your page"


class FacebookBot:
    """
    Бот для отправки сообщений в Facebook с "человеческими" задержками.
    Использует undetected-chromedriver + случайный User-Agent, микроклики и т.д.
    """

    FACEBOOK_LOGIN_URL = "https://www.facebook.com/login"

    def __init__(
        self,
        email: str,
        password: str,
        progress_file: str,
        daily_limit: int,
    ) -> None:
        self.email = email
        self.password = password
        self.progress_file = progress_file
        self.daily_limit = daily_limit

        self.driver = self.create_browser()
        self.success_count = 0
        self.error_count = 0

        # Пул сообщений
        self.company_messages = self.prepare_facebook_messages()

    def create_browser(self) -> uc.Chrome:
        """Создаёт undetected-chromedriver с случайным User-Agent (корректным образом)."""
        user_agent = random.choice(USER_AGENTS)
        logging.info(f"[{self.email}] Выбран User-Agent: {user_agent}")

        options = uc.ChromeOptions()
        options.add_argument("--disable-blink-features=AutomationControlled")
        options.add_argument("--disable-notifications")
        # ВАЖНО: с двумя дефисами "--user-agent="
        options.add_argument(f"--user-agent={user_agent}")
        options.add_argument("--start-maximized")

        driver = uc.Chrome(options=options)
        return driver
```

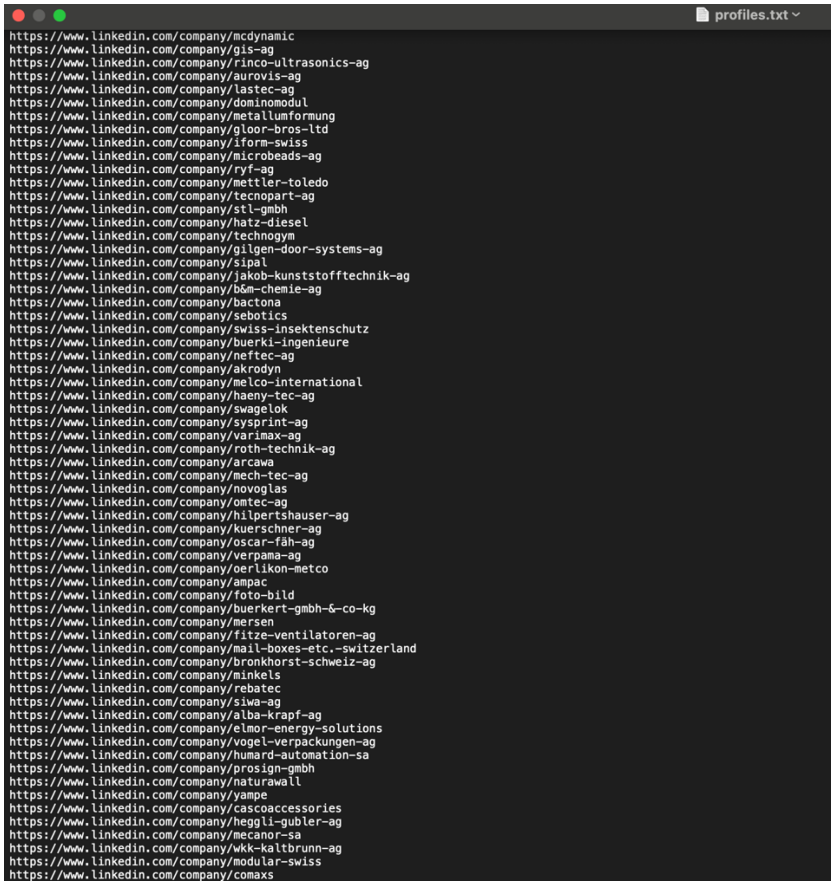
```python
def prepare_facebook_messages(self) -> List[str]:
    """
    20 шаблонов «профессиональных» сообщений, где подставляем {pageName}.
    """
    return [
        "Good day! This is Nikita, Senior Sales Manager at LOG-IN CZ s.r.o. We specialize in secure logistics solutions tailored for high-value cargo. I'd appreciate an opportunity to explore potent
        "Hello, Nikita here from LOG-IN CZ s.r.o. We offer professional and secure logistics solutions specifically designed for companies like {pageName}. Could we arrange a time to discuss this fu
        "Greetings! Nikita from LOG-IN CZ s.r.o. reaching out. Our company provides specialized transport services for valuable and sensitive goods. I believe we can offer effective logistics soluti
        "Good afternoon! Nikita, Senior Sales Manager at LOG-IN CZ s.r.o., here. We deliver highly secure and efficient logistics tailored to your unique business needs. Would {pageName} be open to
        "Hi there, this is Nikita from LOG-IN CZ s.r.o. Our expertise is secure transportation of high-value goods across Europe. I'm keen to discuss how we could support {pageName} in your logistic
        "Hello, Nikita here, representing LOG-IN CZ s.r.o. We specialize in logistics for valuable and bulky cargo, ensuring security and reliability at every stage. Could we schedule a conversation
        "Good day! Nikita from LOG-IN CZ s.r.o. contacting you. Our tailored logistics solutions have successfully supported businesses similar to {pageName}. Would you be interested in exploring a
        "Hello, this is Nikita from LOG-IN CZ s.r.o. We pride ourselves on delivering secure logistics services for high-value products. I think our solutions could greatly benefit {pageName}. Would
        "Greetings! Nikita here from LOG-IN CZ s.r.o. Our dedicated logistics services focus on the safe transportation of valuable goods. I'd like to propose a discussion about potential cooperatio
        "Good afternoon! This is Nikita, Senior Sales Manager at LOG-IN CZ s.r.o. We specialize in reliable logistics for high-value and sensitive cargo, and I would appreciate exploring how our ser
        "Hello, Nikita from LOG-IN CZ s.r.o. We have proven logistics solutions designed specifically for businesses handling high-value cargo like {pageName}. Would you be interested in a brief con
        "Good day! Nikita, Senior Sales Manager at LOG-IN CZ s.r.o., here. We offer comprehensive logistics services tailored for valuable goods. Could we briefly discuss how our expertise could ben
        "Hi, Nikita from LOG-IN CZ s.r.o. We specialize in secure and efficient transport for businesses like {pageName}. Could we explore a potential partnership together? Visit https://www.login-l
        "Good afternoon, Nikita here from LOG-IN CZ s.r.o. Our logistics expertise ensures the secure handling and transportation of high-value products. I would be glad to discuss cooperation oppor
        "Hello, this is Nikita from LOG-IN CZ s.r.o. We excel at delivering secure logistics tailored to specific business needs. I'd like to discuss how we might support {pageName}. Interested in a
        "Greetings! Nikita from LOG-IN CZ s.r.o. here. We are experts in the logistics of valuable cargo, and I'd welcome the chance to explore potential collaboration with {pageName}. Please visit
        "Good day, Nikita speaking from LOG-IN CZ s.r.o. Our logistics services focus specifically on secure transport for high-value goods. Could we schedule a short discussion about working togeth
        "Hello! Nikita, Senior Manager at LOG-IN CZ s.r.o. here. We're dedicated to providing secure and professional logistics services. I'd appreciate an opportunity to discuss how we could help {
        "Hi, Nikita from LOG-IN CZ s.r.o. reaching out. We provide reliable logistics solutions for companies dealing with valuable cargo, like {pageName}. Would you be interested in discussing this
        "Good day! Nikita from LOG-IN CZ s.r.o. here. Our specialized logistics solutions are designed to secure high-value shipments. Could we arrange a discussion about potential collaboration wit
    ]

# ------------------------------
# Методы имитации «человечности»
# ------------------------------
def random_sleep(self, min_sec: float = 2, max_sec: float = 6) -> None:
    """Небольшая случайная пауза (2–6 секунд)."""
    delay = random.uniform(min_sec, max_sec)
    logging.info(f"[{self.email}] Задержка ~{delay:.2f} с...")
    time.sleep(delay)

def random_microclick(self, times: int = 1) -> None:
    """Несколько «микрокликов» в случайных местах окна."""
    action = ActionChains(self.driver)
    try:
        width = self.driver.execute_script("return window.innerWidth")
        height = self.driver.execute_script("return window.innerHeight")
    except:
        return

    for _ in range(times):
        x_offset = random.randint(50, width - 50)
        y_offset = random.randint(50, height - 50)
        try:
            action.move_by_offset(x_offset, y_offset).click().perform()
            action.move_by_offset(-x_offset, -y_offset)  # сбрасываем смещение
        except:
            pass
        self.random_sleep(0.3, 1.0)

def random_mouse_move(self, times: int = 1) -> None:
    """Лёгкое перемещение мыши."""
    action = ActionChains(self.driver)
    for _ in range(times):
        move_x = random.randint(-50, 50)
        move_y = random.randint(-50, 50)
        action.move_by_offset(move_x, move_y).pause(random.uniform(0.2, 0.7))
    try:
        action.perform()
    except:
        pass




def simulate_additional_activity(self) -> None:
    """Случайные движения и клики."""
    if random.choice([True, False]):
        self.random_mouse_move(random.randint(1, 3))
    if random.choice([True, False]):
        self.random_microclick(random.randint(1, 2))
    self.random_sleep(1, 3)

def simulate_page_behavior(self, min_scrolls: int = 2, max_scrolls: int = 5) -> None:
    """Череда скроллов и «микродействий» на странице."""
    scroll_count = random.randint(min_scrolls, max_scrolls)
    logging.info(f"[{self.email}] Имитация поведения: {scroll_count} скроллов.")
    for _ in range(scroll_count):
        scroll_distance = random.randint(300, 1000)
        self.driver.execute_script(f"window.scrollBy(0, {scroll_distance});")
        self.simulate_additional_activity()
        if random.choice([True, False]):
            back_distance = random.randint(50, scroll_distance // 2)
            self.driver.execute_script(f"window.scrollBy(0, {-back_distance});")
            self.random_sleep(1, 3)

# ------------------------------
# Основные методы
# ------------------------------
def accept_cookies(self) -> None:
    """Кликает по кнопке 'Разрешить все cookie' (если найдена)."""
    try:
        cookie_button = WebDriverWait(self.driver, 5).until(
            EC.element_to_be_clickable((By.XPATH, "//span[text()='Разрешить все cookie']"))
        )
        self.scroll_and_click(cookie_button)
        logging.info(f"[{self.email}] Куки приняты.")
        self.random_sleep(1, 3)
    except TimeoutException:
        logging.info(f"[{self.email}] Кнопка принятия cookie не найдена.")
    except Exception:
        logging.exception(f"[{self.email}] Ошибка при нажатии куки.")

def login(self) -> None:
    """Авторизация на Facebook с 60 сек. паузой для 2FA."""
    logging.info(f"[{self.email}] Вход в Facebook...")
    self.driver.get(self.FACEBOOK_LOGIN_URL)
    self.random_sleep(3, 6)

    try:
        self.accept_cookies()
        self.random_sleep(2, 5)

        email_input = self.driver.find_element(By.ID, "email")
        password_input = self.driver.find_element(By.ID, "pass")
        email_input.send_keys(self.email)
        self.random_sleep(1, 3)
        password_input.send_keys(self.password)
        password_input.send_keys(Keys.RETURN)

        logging.info(f"[{self.email}] Данные отправлены. Ждём 60 сек на подтверждение...")
        time.sleep(60)
        logging.info(f"[{self.email}] Предполагаем, что авторизация успешна.")
    except Exception:
        logging.exception(f"[{self.email}] Ошибка при входе.")
        self.error_count += 1

def scroll_and_click(self, element) -> None:
    """Прокрутить к элементу и кликнуть (JS, затем ActionChains)."""
    try:
        self.driver.execute_script("arguments[0].scrollIntoView({block: 'center'});", element)
        self.random_sleep(1, 2)
```

```
            self.random_sleep(1, 2)
            self.driver.execute_script("arguments[0].click();", element)
            self.random_sleep(1, 2)
        except:
            logging.exception(f"[{self.email}] Ошибка JS-клика, пробуем ActionChains.")
            try:
                ActionChains(self.driver).move_to_element(element).pause(1).click(element).perform()
                self.random_sleep(1, 2)
            except Exception:
                logging.exception(f"[{self.email}] Ошибка ActionChains-клика.")

    def go_to_profile(self, profile_url: str) -> bool:
        """Открывает профиль/страницу и имитирует действия."""
        try:
            self.driver.get(profile_url)
            self.random_sleep(5, 12)
            self.simulate_page_behavior()

            current_url = self.driver.current_url.lower()
            if "login" in current_url:
                logging.info(f"[{self.email}] Снова требует логин: {profile_url}")
                return False
            logging.info(f"[{self.email}] Открыт профиль: {profile_url}")
            return True
        except Exception:
            logging.exception(f"[{self.email}] Ошибка при переходе: {profile_url}.")
            self.error_count += 1
            return False

    def close_chat(self) -> None:
        """Закрывает чат, отправляя Escape."""
        try:
            ActionChains(self.driver).send_keys(Keys.ESCAPE).perform()
            logging.info(f"[{self.email}] Чат закрыт (Escape).")
            self.random_sleep(1, 2)
        except Exception:
            logging.exception(f"[{self.email}] Ошибка при закрытии чата.")

    def partial_type_message(self, element, text: str) -> None:
        """Посимвольный ввод текста."""
        for ch in text:
            element.send_keys(ch)
            time.sleep(random.uniform(0.05, 0.15))

    def send_message(self, profile_url: str) -> Union[bool, str]:
        """
        Отправляет сообщение в чат:
         - True: успешно
         - "invalid": нет профиля / нет кнопки
         - False: ошибка
        """
        if not self.go_to_profile(profile_url):
            return "invalid"

        # Находим кнопку "Message" (англ. интерфейс)
        try:
            message_button = WebDriverWait(self.driver, 10).until(
                EC.element_to_be_clickable((By.XPATH, "//span[text()='Message']"))
            )
            self.scroll_and_click(message_button)
            self.random_sleep(3, 6)
        except TimeoutException:
            logging.info(f"[{self.email}] Кнопка 'Message' не найдена: {profile_url}.")
            return "invalid"
        except Exception:
            logging.exception(f"[{self.email}] Ошибка при нажатии 'Message'.")
            self.error_count += 1
            return False



            return False

        # Находим поле ввода сообщения
        try:
            text_input = WebDriverWait(self.driver, 10).until(
                EC.presence_of_element_located((By.XPATH, "//p[@class='xat24cr xdj266r']"))
            )
            message = self.get_company_message(profile_url)
            self.partial_type_message(text_input, message)
            self.random_sleep(1, 3)

            text_input.send_keys(Keys.ENTER)
            logging.info(f"[{self.email}] Сообщение отправлено: {profile_url}")
            self.success_count += 1

            self.random_sleep(1, 2)
            self.close_chat()
            return True

        except TimeoutException:
            logging.error(f"[{self.email}] Поле сообщения не найдено: {profile_url}.")
            return False
        except Exception:
            logging.exception(f"[{self.email}] Ошибка при вводе/отправке: {profile_url}.")
            self.error_count += 1
            return False

    def get_company_message(self, profile_url: str) -> str:
        """Подставляем {pageName} в один из случайных шаблонов."""
        page_name = extract_page_name(profile_url)
        template = random.choice(self.company_messages)
        return template.replace("{pageName}", page_name)

    def load_profiles(self) -> List[str]:
        """Читаем список профилей из файла fb_profiles.txt."""
        p = Path(PROFILES_FILE_FB)
        if not p.exists():
            logging.error(f"[{self.email}] Файл {p} не найден!")
            return []
        try:
            with p.open("r", encoding="utf-8") as f:
                profiles = [line.strip() for line in f if line.strip()]
            logging.info(f"[{self.email}] Загружено {len(profiles)} ссылок из {p}.")
            return profiles
        except Exception:
            logging.exception(f"[{self.email}] Ошибка чтения профилей.")
            self.error_count += 1
            return []


def load_progress_fb(progress_file: str) -> int:
    """Считываем индекс профиля из JSON."""
    p = Path(progress_file)
    if not p.exists():
        return 0
    try:
        data = json.loads(p.read_text(encoding="utf-8"))
        return data.get("current_index", 0)
    except:
        return 0


def save_progress_fb(progress_file: str, current_index: int) -> None:
    """Сохраняем текущий индекс профиля."""
    data = {"current_index": current_index}
    Path(progress_file).write_text(json.dumps(data, ensure_ascii=False), encoding="utf-8")
```

```
    Path(progress_file).write_text(json.dumps(data, ensure_ascii=False), encoding="utf-8")


def daily_run_facebook(bot: FacebookBot) -> None:
    """
    Разовая задача рассылки (ограничена WORK_START–WORK_END).
    Переходит по списку профилей и отправляет сообщения.
    """
    logging.info(f"=== Начало дневной задачи: [{bot.email}] ===")
    profiles = bot.load_profiles()
    if not profiles:
        logging.error(f"[{bot.email}] Нет профилей для обработки.")
        return

    current_index = load_progress_fb(bot.progress_file)
    logging.info(f"[{bot.email}] Текущий индекс: {current_index}")
    sent_today = 0

    while True:
        now = datetime.now()
        if now.hour >= WORK_END:
            logging.info(f"[{bot.email}] Время >= {WORK_END}:00. Останавливаем работу.")
            break

        if sent_today >= bot.daily_limit:
            logging.info(f"[{bot.email}] Достигнут лимит {bot.daily_limit} сообщений.")
            break

        if current_index >= len(profiles):
            logging.info(f"[{bot.email}] Профили закончились.")
            break

        profile_url = profiles[current_index]
        logging.info(f"[{bot.email}] Отправка профилю #{current_index}: {profile_url}")

        result = bot.send_message(profile_url)
        if result is True:
            sent_today += 1
            # Пауза 30–90 секунд
            pause_seconds = random.randint(30, 90)
            logging.info(f"[{bot.email}] Ждём {pause_seconds} сек перед следующим профилем.")
            time.sleep(pause_seconds)
        elif result == "invalid":
            logging.info(f"[{bot.email}] Ссылка невалидна/кнопка не найдена. Пропуск без задержки.")
        else:
            logging.info(f"[{bot.email}] Ошибка при отправке. Пропуск без задержки.")

        current_index += 1
        save_progress_fb(bot.progress_file, current_index)

    # Итоги
    logging.info(f"=== Итог за сегодня (аккаунт {bot.email}) ===")
    logging.info(f"[{bot.email}] Отправлено сегодня: {sent_today}")
    logging.info(f"[{bot.email}] Успешно всего: {bot.success_count}")
    logging.info(f"[{bot.email}] Ошибок всего: {bot.error_count}")
    logging.info(f"=== Завершение daily_run_facebook [{bot.email}] ===\n")


def main() -> None:
    """
    Основная функция — создаёт 2 бота, логинится, планирует рассылку (06:00),
    а также выполняет рассылку сразу, если текущее время в промежутке 6–17.
    """
    EMAIL_1 = "produktdum@gmail.com"
    PASSWORD_1 = "Qwer114214"
    bot1 = FacebookBot(
        email=EMAIL_1,
        password=PASSWORD_1.


       progress_file=PROGRESS_FILE_FB_1,
       daily_limit=DAILY_LIMIT_1
    )

    EMAIL_2 = "batonrus26@gmail.com"
    PASSWORD_2 = "Cfifnikita2003?010"
    bot2 = FacebookBot(
        email=EMAIL_2,
        password=PASSWORD_2,
        progress_file=PROGRESS_FILE_FB_2,
        daily_limit=DAILY_LIMIT_2
    )

    # Логинимся (с паузой между аккаунтами, чтобы не делать всё разом)
    bot1.login()
    logging.info("[MAIN] Пауза перед логином второго аккаунта...")
    time.sleep(random.randint(30, 60))

    bot2.login()
    logging.info("[MAIN] Пауза после второго логина...")
    time.sleep(random.randint(30, 60))

    # Планируем задачу рассылки на 06:00 для обоих аккаунтов
    schedule.every().day.at("06:00").do(daily_run_facebook, bot=bot1)
    schedule.every().day.at("06:00").do(daily_run_facebook, bot=bot2)

    # Если сейчас между 6:00 и 17:00, запускаем рассылку немедленно
    current_hour = datetime.now().hour
    if WORK_START <= current_hour < WORK_END:
        logging.info("[MAIN] Сейчас рабочее время. Запускаем рассылку сразу.")
        daily_run_facebook(bot1)
        time.sleep(random.randint(20, 40))
        daily_run_facebook(bot2)
    else:
        logging.info(f"[MAIN] Время вне диапазона {WORK_START}–{WORK_END}. Ждём 06:00.")

    try:
        while True:
            schedule.run_pending()
            time.sleep(30)
    except KeyboardInterrupt:
        logging.info("[MAIN] Завершение по Ctrl+C.")
    finally:
        bot1.driver.quit()
        bot2.driver.quit()
        logging.info("[MAIN] Браузеры закрыты. Работа завершена.")

if __name__ == "__main__":
    main()
```

Zdroj: Autor

Příloha F profiles.txt



Zdroj: Autor

Příloha G email_validator.py

```
import os
import logging
import random
import string
import time
from typing import List, Tuple, Dict, Optional
import datetime
import socket
import smtplib

import dns.resolver
import dns.reversename
import spf
import dkim
from validate_email_address import validate_email
import openpyxl
from openpyxl import Workbook
from openpyxl.worksheet.worksheet import Worksheet
from email.message import EmailMessage

from concurrent.futures import ThreadPoolExecutor, as_completed

# ------------------------------------------------------------------------------
#                              НАСТРОЙКИ ПОЛЬЗОВАТЕЛЯ
# ------------------------------------------------------------------------------

# ========== Базовые пути ==========
# Имя основного Excel-файла (без даты). РЕАЛЬНЫЙ ИСХОДНЫЙ ФАЙЛ: emails_validated.xlsx
# При включённом ARCHIVE_RESULTS он будет копироваться с датой в имени.
EXCEL_FILE_BASE = "emails_validated"

# Имя листа, где хранятся email-адреса
SHEET_NAME = "Sheet1"

# Номера строк и колонок
START_ROW = 2                  # Первая строка с данными
EMAIL_COL = 2                  # Колонка B
STATUS_COL = 4                 # Колонка D
REASON_COL = 5                 # Колонка E

# Файл для хранения чекпоинта (номер последней обработанной строки)
CHECKPOINT_FILE = "checkpoint.txt"

# ========== Логи ==========
LOG_DIR = "logs"               # Директория для логов
if not os.path.exists(LOG_DIR):
    os.makedirs(LOG_DIR)

# Лог-файл с датой в названии
LOG_FILE = os.path.join(LOG_DIR, f"log_{datetime.date.today()}.log")

# ========== SMTP-отправка отчёта ==========
SMTP_SERVER = "smtp.gmail.com"
SMTP_PORT = 587
# Чтобы не хранить чувствительные данные в коде, попробуйте использовать переменные окружения:
#   export MY_SMTP_USER="mylogin@example.com"
#   export MY_SMTP_PASS="app_password"
SMTP_USER = os.getenv("MY_SMTP_USER", "nikitasasenko@gmail.com")
SMTP_PASS = os.getenv("MY_SMTP_PASS", "hgiceqtchvstoczw")

# Куда отправлять отчёт (по умолчанию — на тот же адрес)
SEND_REPORT_TO = SMTP_USER

# ========== Основные режимы ==========
DETECT_CATCH_ALL = True
CATCH_ALL_LABEL = "CATCH_ALL"
# Если домен оказался catch-all, не помечаем как INVALID, а ставим отдельный статус



# Повторная проверка при greylisting (4xx)
GREYLIST_RETRY_WAIT = 120  # 2 минуты

# Лимит строк за сутки
DAILY_LIMIT = 100
# Ждать ли до полуночи, если остались строки, но достигнут DAILY_LIMIT
WAIT_UNTIL_MIDNIGHT = True

# Надо ли повторно проверять ERROR-адреса (коды 5xx или иные)
RECHECK_ERRORS = True
RECHECK_WAIT = 10  # секунды перед повторной проверкой

# ========== Паузы и параллелизм ==========
MIN_PAUSE = 1     # Случайная задержка между проверками, сек.
MAX_PAUSE = 3
MAX_WORKERS = 4   # Количество потоков при параллельной обработке

# ========== Spamhaus / SPF / DKIM / DMARC ==========
CHECK_SPAMHAUS = True
SPAMHAUS_ZEN = "zen.spamhaus.org"

CHECK_SPF = True
CHECK_DKIM = True
CHECK_DMARC = True

# ========== Архивирование результатов ==========
ARCHIVE_RESULTS = True
ARCHIVE_DIR = "results_archive"
if ARCHIVE_RESULTS and not os.path.exists(ARCHIVE_DIR):
    os.makedirs(ARCHIVE_DIR)

# ------------------------------------------------------------------------------
#                              НАСТРОЙКА ЛОГИРОВАНИЯ
# ------------------------------------------------------------------------------
def setup_logging() -> None:
    """Инициализируем logging: вывод в файл + консоль."""
    logging.basicConfig(
        level=logging.INFO,
        format="%(asctime)s [%(levelname)s] %(message)s",
        datefmt="%Y-%m-%d %H:%M:%S",
        handlers=[
            logging.FileHandler(LOG_FILE, mode="a", encoding='utf-8'),
            logging.StreamHandler()
        ]
    )

# ------------------------------------------------------------------------------
#                           ВСПОМОГАТЕЛЬНЫЕ ФУНКЦИИ
# ------------------------------------------------------------------------------
def basic_format_check(email: str) -> bool:
    """
    Базовая проверка формата email с помощью validate_email_address.
    """
    try:
        return bool(validate_email(email, verify=False))
    except:
        return False

def random_pause():
    """
    Случайная пауза между MIN_PAUSE и MAX_PAUSE.
    """
    delay = random.uniform(MIN_PAUSE, MAX_PAUSE)
    logging.debug(f"Пауза {delay:.2f} сек.")
    time.sleep(delay)

def get_checkpoint() -> int:
```

```
    Возвращает номер последней обработанной строки (или START_ROW-1, если нет файла).
    """
    try:
        with open(CHECKPOINT_FILE, "r", encoding="utf-8") as f:
            return int(f.read().strip())
    except:
        return START_ROW - 1

def save_checkpoint(row_index: int):
    """
    Сохраняет номер последней обработанной строки в файл.
    """
    with open(CHECKPOINT_FILE, "w", encoding="utf-8") as f:
        f.write(str(row_index))

def wait_until_midnight():
    """
    Ожидаем до следующей полуночи (если включен WAIT_UNTIL_MIDNIGHT).
    """
    if not WAIT_UNTIL_MIDNIGHT:
        return
    now = time.localtime()
    seconds_until_midnight = (
        (23 - now.tm_hour) * 3600 +
        (59 - now.tm_min) * 60 +
        (60 - now.tm_sec)
    )
    logging.info(f"Достигнут дневной лимит. Ждём {seconds_until_midnight} сек. до новой суток.")
    time.sleep(seconds_until_midnight)

# ------------------------------------------------------------------------
#                            DNS И SMTP-ПРОВЕРКИ
# ------------------------------------------------------------------------
MX_RECORD_CACHE: Dict[str, List[Tuple[str, int]]] = {}
SMTP_CACHE: Dict[str, Tuple[str, str]] = {}  # email -> (status, reason)
CATCH_ALL_DOMAIN_CACHE: Dict[str, bool] = {}

def check_port_25(mx_host: str) -> bool:
    """
    Проверяем, открыт ли порт 25 у данного MX-хоста.
    Если закрыт - возвращаем False.
    """
    try:
        with socket.create_connection((mx_host, 25), timeout=5):
            return True
    except:
        return False

def get_mx_records(domain: str) -> List[Tuple[str, int]]:
    """
    Возвращаем список MX-записей [(mx_host, preference)], отсортированных по приоритету.
    """
    if domain in MX_RECORD_CACHE:
        return MX_RECORD_CACHE[domain]

    try:
        answers = dns.resolver.resolve(domain, 'MX')
        records = [(str(r.exchange).rstrip('.'), r.preference) for r in answers]
        sorted_records = sorted(records, key=lambda x: x[1])
        MX_RECORD_CACHE[domain] = sorted_records
        return sorted_records
    except:
        MX_RECORD_CACHE[domain] = []
        return []

def spamhaus_check(domain: str) -> bool:
    """
    Проверка IP-адреса MX в Spamhaus (zen.spamhaus.org).



    """
    Проверка IP-адреса MX в Spamhaus (zen.spamhaus.org).
    Если IP найден в чёрном списке - возвращаем True.
    """
    if not CHECK_SPAMHAUS:
        return False

    mx_list = get_mx_records(domain)
    if not mx_list:
        return False

    # Берём первый MX
    mx_host = mx_list[0][0]
    try:
        ip = socket.gethostbyname(mx_host)
        rev_ip = ".".join(ip.split(".")[::-1])
        query = f"{rev_ip}.{SPAMHAUS_ZEN}"
        dns.resolver.resolve(query, "A")  # если не упадёт - IP в списке
        return True
    except dns.resolver.NXDOMAIN:
        return False
    except:
        return False

def check_spf_record(domain: str) -> bool:
    """
    Упрощённая проверка SPF: ищем TXT с 'v=spf1', затем используем pyspf (spf.query) на нейтральном IP.
    """
    if not CHECK_SPF:
        return False
    try:
        txt_records = dns.resolver.resolve(domain, 'TXT')
        for rdata in txt_records:
            txt_str = rdata.to_text().strip('"')
            if txt_str.startswith("v=spf1"):
                # Пытаемся более-менее проверить
                res = spf.query(i="1.2.3.4", s=f"test@{domain}", h=domain).check()
                # res[0] может быть "pass", "neutral", "softfail" и т.д.
                # Считаем OK, если pass или neutral
                return res[0] in ("pass", "neutral")
        return False
    except:
        return False

def check_dkim_record(domain: str) -> bool:
    """
    Упрощённая проверка наличия DNS-записи default._domainkey.<domain>.
    Это не полноценная верификация DKIM-подписи письма.
    """
    if not CHECK_DKIM:
        return False
    try:
        dkim_selector = "default._domainkey"
        full_dkim_domain = f"{dkim_selector}.{domain}"
        txt_records = dns.resolver.resolve(full_dkim_domain, 'TXT')
        if txt_records:
            return True
        return False
    except:
        return False

def check_dmarc_record(domain: str) -> bool:
    """
    Упрощённая проверка DMARC: наличие _dmarc.<domain> с "v=DMARC1".
    """
    if not CHECK_DMARC:
        return False
    try:
```

```
            return False
        try:
            dmarc_domain = f"_dmarc.{domain}"
            txt_records = dns.resolver.resolve(dmarc_domain, 'TXT')
            for r in txt_records:
                txt_str = r.to_text().strip('"')
                if txt_str.startswith("v=DMARC1"):
                    return True
            return False
        except:
            return False

    def generate_fake_localpart(length: int = 8) -> str:
        """
        Генерируем случайную локальную часть, чтобы проверить catch-all.
        """
        chars = string.ascii_lowercase + string.digits
        return ''.join(random.choice(chars) for _ in range(length))

    def detect_catch_all(domain: str) -> bool:
        """
        Проверяем, является ли домен catch-all путём отправки запроса на несуществующий адрес.
        """
        if domain in CATCH_ALL_DOMAIN_CACHE:
            return CATCH_ALL_DOMAIN_CACHE[domain]

        fake_email = generate_fake_localpart() + "@" + domain
        status, _ = smtp_check_email(fake_email, skip_cache=True)
        is_catch_all = (status == "VALID")
        CATCH_ALL_DOMAIN_CACHE[domain] = is_catch_all
        return is_catch_all

    def smtp_check_email(email_address: str, skip_cache: bool = False) -> Tuple[str, str]:
        """
        Проверка email через SMTP: возвращаем (status, reason).
        - status: VALID / INVALID / ERROR
        - reason: краткое описание, включая код ответа сервера
        """
        # Если в кэше есть, отдаём (если skip_cache=False)
        if not skip_cache and email_address in SMTP_CACHE:
            return SMTP_CACHE[email_address]

        if "@" not in email_address:
            res = ("INVALID", "Bad format (no @)")
            if not skip_cache:
                SMTP_CACHE[email_address] = res
            return res

        local_part, domain = email_address.split("@", 1)
        domain = domain.lower().strip()

        # Получаем MX
        mx_list = get_mx_records(domain)
        if not mx_list:
            res = ("INVALID", "No MX records")
            if not skip_cache:
                SMTP_CACHE[email_address] = res
            return res

        for mx_host, _pref in mx_list:
            # Сначала проверим порт 25
            if not check_port_25(mx_host):
                logging.debug(f"Port 25 closed on {mx_host}")
                continue
            try:
                logging.info(f"[SMTP] Подключение к {mx_host} для {email_address}")
                with smtplib.SMTP(mx_host, 25, timeout=10) as server:
                    server.ehlo or helo if needed()


                    code, msg = server.mail("")
                    code, msg = server.rcpt(email_address)

                    msg_decoded = (
                        msg.decode(errors="ignore") if hasattr(msg, "decode") else str(msg)
                    )

                    if 200 <= code < 300:
                        res = ("VALID", f"{code} {msg_decoded}")
                    elif 400 <= code < 500:
                        # greylisting / временные ошибки
                        res = ("ERROR", f"{code} {msg_decoded}")
                    elif 500 <= code < 600:
                        res = ("INVALID", f"{code} {msg_decoded}")
                    else:
                        res = ("ERROR", f"{code} {msg_decoded}")

                    # Сохраняем в кэш (если skip_cache=False)
                    if not skip_cache:
                        SMTP_CACHE[email_address] = res
                    return res

            except Exception as e:
                logging.warning(f"Ошибка подключения к {mx_host}: {e}")
                continue

        # Если все MX недоступны
        res = ("ERROR", "No working MX or connection failed")
        if not skip_cache:
            SMTP_CACHE[email_address] = res
        return res

    # ------------------------------------------------------------------------
    #                       ОСНОВНАЯ ЛОГИКА ПРОВЕРКИ
    # ------------------------------------------------------------------------
    def validate_email_row(row_index: int, sheet: Worksheet) -> None:
        """
        Проверяем один email (в одной строке Excel):
        1) Формат
        2) Spamhaus
        3) SPF/DKIM/DMARC (просто для справки, не меняем VALID/INVALID)
        4) Catch-all (если включено)
        5) SMTP-проверка, с возможным greylisting retry
        """
        email_cell_val = sheet.cell(row=row_index, column=EMAIL_COL).value
        if not email_cell_val:
            return

        email_str = str(email_cell_val).strip()
        logging.info(f"[Row {row_index}] Проверка: {email_str}")

        # 1) Базовая проверка формата
        if not basic_format_check(email_str):
            sheet.cell(row=row_index, column=STATUS_COL).value = "INVALID"
            sheet.cell(row=row_index, column=REASON_COL).value = "Format error"
            return

        # 2) Spamhaus
        domain = email_str.split("@", 1)[1].lower()
        if spamhaus_check(domain):
            sheet.cell(row=row_index, column=STATUS_COL).value = "INVALID"
            sheet.cell(row=row_index, column=REASON_COL).value = "Spamhaus"
            return

        # 3) SPF / DKIM / DMARC (доп. сведения)
        spf_ok = check_spf_record(domain)
        dkim_ok = check_dkim_record(domain)
        dmarc_ok = check_dmarc_record(domain)
```

```
    dkim_ok = check_dkim_record(domain)
    dmarc_ok = check_dmarc_record(domain)

    # 4) Catch-all
    if DETECT_CATCH_ALL:
        is_catch_all = detect_catch_all(domain)
        if is_catch_all:
            sheet.cell(row=row_index, column=STATUS_COL).value = CATCH_ALL_LABEL
            sheet.cell(row=row_index, column=REASON_COL).value = (
                f"Catch-all domain. SPF={spf_ok}, DKIM={dkim_ok}, DMARC={dmarc_ok}"
            )
            return

    # 5) SMTP-проверка
    status, reason = smtp_check_email(email_str)
    if status == "ERROR" and reason.startswith("4"):  # greylisting / 4xx
        logging.info(f"Greylisting suspected: повторим через {GREYLIST_RETRY_WAIT} сек.")
        time.sleep(GREYLIST_RETRY_WAIT)
        # Повтор без кэша
        status, reason = smtp_check_email(email_str, skip_cache=True)

    reason_extra = f"{reason} | SPF={spf_ok},DKIM={dkim_ok},DMARC={dmarc_ok}"
    sheet.cell(row=row_index, column=STATUS_COL).value = status
    sheet.cell(row=row_index, column=REASON_COL).value = reason_extra

    # Случайная пауза
    random_pause()


def validate_batch(sheet: Worksheet, row_indices: List[int]) -> List[int]:
    """
    Обрабатываем группу строк (параллельно в ThreadPoolExecutor).
    Возвращаем список индексов, где статус = ERROR (если нужен повтор).
    """
    need_recheck = []
    with ThreadPoolExecutor(max_workers=MAX_WORKERS) as executor:
        future_map = {}
        for row_idx in row_indices:
            fut = executor.submit(validate_email_row, row_idx, sheet)
            future_map[fut] = row_idx

        for fut in as_completed(future_map):
            row_idx = future_map[fut]
            try:
                fut.result()
                # После выполнения проверки смотрим, что получилось
                status_val = sheet.cell(row=row_idx, column=STATUS_COL).value
                if status_val == "ERROR":
                    need_recheck.append(row_idx)
            except Exception as e:
                logging.error(f"Ошибка валидации для строки {row_idx}: {e}")

    return need_recheck

def recheck_errors(sheet: Worksheet, row_indices: List[int]) -> None:
    """
    Повторно проверяем адреса со статусом ERROR, через RECHECK_WAIT сек.
    """
    if not row_indices:
        return
    logging.info(f"RECHECK {len(row_indices)} адресов в статусе ERROR...")
    time.sleep(RECHECK_WAIT)
    with ThreadPoolExecutor(max_workers=MAX_WORKERS) as executor:
        future_map = {}
        for row_idx in row_indices:
            fut = executor.submit(validate_email_row, row_idx, sheet)
            future_map[fut] = row_idx




        for fut in as_completed(future_map):
            row_idx = future_map[fut]
            try:
                fut.result()
            except Exception as e:
                logging.error(f"Ошибка повторной проверки для строки {row_idx}: {e}")

def build_summary(sheet: Worksheet, up_to_row: int) -> str:
    """
    Собираем статистику: сколько VALID, INVALID, ERROR, CATCH_ALL.
    """
    from collections import Counter
    statuses = []
    for r in range(START_ROW, up_to_row + 1):
        val = sheet.cell(row=r, column=STATUS_COL).value
        if val:
            statuses.append(val)

    c = Counter(statuses)
    lines = []
    for st in ["VALID", "INVALID", "ERROR", CATCH_ALL_LABEL]:
        if c[st] > 0:
            lines.append(f"{st}: {c[st]}")

    if not lines:
        return "Пока нет данных о статусах."

    return "\n".join(lines)

# ------------------------------------------------------------------------
#                ОТПРАВКА ПИСЬМА С ИТОГОВЫМ (ИЛИ ПРОМЕЖУТОЧНЫМ) ОТЧЁТОМ
# ------------------------------------------------------------------------
def send_report_email(excel_filename: str, summary_text: str) -> None:
    """
    Отправляем письмо с вложенным Excel-файлом + статистикой в теле письма.
    """
    msg = EmailMessage()
    msg["Subject"] = f"✅ Результат проверки email-ов {datetime.date.today()}"
    msg["From"] = SMTP_USER
    msg["To"] = SEND_REPORT_TO

    msg.set_content(
        f"Здравствуйте!\n\n"
        f"Во вложении — актуальный файл с проверенными email.\n\n"
        f"Краткая статистика:\n{summary_text}\n\n"
        f"—\n"
        f"Это автоматически сгенерированное сообщение."
    )

    try:
        with open(excel_filename, "rb") as f:
            msg.add_attachment(
                f.read(),
                maintype="application",
                subtype="vnd.openxmlformats-officedocument.spreadsheetml.sheet",
                filename=os.path.basename(excel_filename)
            )

        with smtplib.SMTP(SMTP_SERVER, SMTP_PORT) as server:
            server.starttls()
            server.login(SMTP_USER, SMTP_PASS)
            server.send_message(msg)

        logging.info("✅ Письмо с отчётом отправлено.")
    except Exception as e:
        logging.error(f"Ошибка при отправке письма: {e}")
```

```
        logging.error(f"Ошибка при отправке письма: {e}")

# ------------------------------------------------------------------------
#                                   MAIN
# ------------------------------------------------------------------------
def main():
    setup_logging()
    logging.info("=== СТАРТ ПРОГРАММЫ ===")

    # Формируем «архивное» имя Excel-файла с датой (emails_validated_2025-04-06.xlsx)
    date_str = datetime.date.today().isoformat()
    excel_with_date = f"{EXCEL_FILE_BASE}_{date_str}.xlsx"

    # Если включено архивирование — копируем исходный файл в файл с датой
    if ARCHIVE_RESULTS:
        if os.path.exists(f"{EXCEL_FILE_BASE}.xlsx") and not os.path.exists(excel_with_date):
            import shutil
            shutil.copyfile(f"{EXCEL_FILE_BASE}.xlsx", excel_with_date)
            logging.info(f"Скопирован {EXCEL_FILE_BASE}.xlsx в {excel_with_date}")
        # Рабочим файлом будет архивный
        excel_file_name = excel_with_date
    else:
        # Иначе работаем с исходным
        excel_file_name = f"{EXCEL_FILE_BASE}.xlsx"

    last_processed = get_checkpoint()
    logging.info(f"Начинаем с строки: {last_processed + 1}")

    while True:
        # Открываем Excel
        try:
            wb: Workbook = openpyxl.load_workbook(excel_file_name)
            sheet: Worksheet = wb[SHEET_NAME]
        except Exception as e:
            logging.error(f"Ошибка открытия Excel: {e}")
            return

        max_row = sheet.max_row
        start_row = last_processed + 1
        if start_row > max_row:
            logging.info("Все строки уже обработаны.")
            break

        # Определяем конечную строку (учитывая DAILY_LIMIT)
        if DAILY_LIMIT is not None:
            end_row = min(start_row + DAILY_LIMIT - 1, max_row)
        else:
            end_row = max_row

        logging.info(f"Обработка строк {start_row}..{end_row} из {max_row}")

        row_range = list(range(start_row, end_row + 1))

        # Основная проверка батча
        need_rechk = validate_batch(sheet, row_range)

        # Повторная проверка ERROR
        if RECHECK_ERRORS and need_rechk:
            recheck_errors(sheet, need_rechk)

        # Сохраняем результат
        try:
            wb.save(excel_file_name)
            logging.info(f"Сохранён файл {excel_file_name} (строки {start_row}..{end_row}).")
        except Exception as e:
            logging.error(f"Ошибка при сохранении Excel: {e}")

        # Обновляем чекпоинт



        # Обновляем чекпоинт
        last_processed = end_row
        save_checkpoint(last_processed)
        logging.info(f"Чекпоинт: {last_processed} / {max_row}")

        # Проверяем, достигли конца файла?
        if last_processed >= max_row:
            logging.info("Все строки обработаны.")
            break
        else:
            # Отправляем промежуточный отчёт
            summary_text = build_summary(sheet, last_processed)
            send_report_email(excel_file_name, summary_text)

            # Ждём до полуночи, если включено
            wait_until_midnight()

    # Когда всё готово, формируем и отправляем финальный отчёт
    logging.info("Все строки обработаны. Отправляем финальный отчёт.")
    try:
        wb = openpyxl.load_workbook(excel_file_name)
        sheet = wb[SHEET_NAME]
        summary_text = build_summary(sheet, sheet.max_row)
        wb.save(excel_file_name)
    except:
        summary_text = "Ошибка чтения финального статуса."

    send_report_email(excel_file_name, summary_text)
    logging.info("=== ЗАВЕРШЕНИЕ ПРОГРАММЫ ===")


if __name__ == "__main__":
    main()
```

Zdroj: Autor

Příloha H Legitimate Interest Assessment (LIA) LOG-IN

Legitimate Interest Assessment (LIA)

Společnost: LOG-IN CZ s.r.o.
Datum: 17. 6. 2025
Zpracování osobních údajů na základě oprávněného zájmu dle čl. 6(1)(f) GDPR

1. Identifikace oprávněného zájmu

Společnost LOG-IN CZ s.r.o. má oprávněný zájem oslovovat potenciální firemní klienty (právnické osoby) za účelem nabídky logistických služeb v oblasti B2B přepravy zboží. Získávání nových zákazníků je klíčovým faktorem pro růst společnosti.

2. Posouzení nezbytnosti zpracování

Zpracování základních kontaktních údajů (e-mailové adresy typu info@, obchod@, logistics@) je nezbytné pro realizaci obchodní komunikace. Údaje jsou veřejně dostupné na oficiálních webech firem a zpracovávány pouze v rozsahu nezbytném pro navázání kontaktu (název firmy, e-mailová adresa, popis činnosti).

3. Vyvážení zájmů a práv subjektů údajů

- Jedná se o **firemní e-maily (ne osobní)**, u kterých se dle ÚOOÚ nepředpokládá vysoká míra soukromí.
- E-maily obsahují **jednoduchý opt-out mechanismus** („odpovězte STOP“ nebo „klikněte zde pro odhlášení“).
- Nedochází k použití citlivých údajů ani k automatizovanému rozhodování s právním účinkem (čl. 22 GDPR).
- Všechny aktivity jsou v souladu s principy **„privacy by design“** – systém Make.com zajišťuje filtrování osobních adres, logging, a transparentnost.

Zdroj: Autor, formulář LIA vychází ze vzoru EDPB (2024) a byl přizpůsoben konkrétním podmínkám společnosti LOG-IN CZ

Příloha I setup_server.sh

setup_server.sh

```
#!/bin/bash

echo "⚠️ УДАЛЕНИЕ СТАРЫХ ФАЙЛОВ..."
rm -f /root/email_validator.py
rm -f /root/emails_validated.xlsx
rm -f /root/checkpoint.txt
rm -rf /root/results_archive
rm -rf /root/logs

echo "✅ СТАРЫЕ ФАЙЛЫ УДАЛЕНЫ"

echo "📦 УСТАНОВКА ЗАВИСИМОСТЕЙ..."
apt update && apt install -y python3-pip python3-dev libffi-dev libssl-dev
pip3 install --upgrade pip
pip3 install openpyxl dnspython validate_email_address dkim pyspf

echo "✅ ВСЕ БИБЛИОТЕКИ УСТАНОВЛЕНЫ"

echo "📁 СОЗДАЮ ПАПКИ..."
mkdir -p /root/results_archive
mkdir -p /root/logs

echo "🚀 ГОТОВО! Можешь загружать файлы и запускать скрипт!"
```

Zdroj: Autor