МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»

Кафедра теории функций и стохастического анализа

РАЗРАБОТКА АВТОНОМНОГО ИНТЕЛЛЕКТУАЛЬНОГО АГЕНТА ДЛЯ ВЗАИМОДЕЙСТВИЯ С ВЕБ-СЕРВИСАМИ И ЭЛЕКТРОННОЙ ПОЧТОЙ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

Студента 4 курса 451 группы
направления 38.03.05 — Бизнес-информатика

механико-математического факультета
Князева Олега Александровича

Научный руководитель
доцент, к. ф.-м. н. _____ Д. В. Мельничук

Заведующий кафедрой
д. ф.-м. н., профессор _____ С. П. Сидоров

Саратов 2026

Введение. В условиях цифровой трансформации бизнес-процессов возрастают требования к автоматизации рутинных операций, связанных с взаимодействием с веб-сервисами и системами электронной почты. Традиционные системы роботизированной автоматизации процессов (RPA), такие как UiPath, Blue Prism и Selenium, функционируют на основе жёстко заданных алгоритмов и DOM-селекторов, что обеспечивает высокую скорость выполнения детерминированных операций, однако обладает фундаментальным недостатком — хрупкостью: любое изменение пользовательского интерфейса приводит к поломке сценариев и требует вмешательства разработчика. Кроме того, классические RPA-системы не способны самостоятельно обрабатывать неструктурированные данные, такие как тексты электронных писем, без интеграции с дополнительными модулями OCR или NLP.

С развитием технологий искусственного интеллекта, в частности крупных языковых моделей (LLM), появилась возможность создания автономных интеллектуальных агентов, способных воспринимать окружающую среду через браузер или API, принимать решения на основе естественного языка и выполнять действия с помощью внешних инструментов (Tool Calling). Однако применение языковых моделей в автономном режиме сопряжено с серьёзными рисками: склонность моделей к «галлюцинациям» и уязвимость к состязательным атакам может привести к вызову несуществующих функций, а отсутствие жёсткого контроля создаёт угрозу выполнения деструктивных операций.

Целью данной бакалаврской работы является проектирование и программная реализация автономного интеллектуального агента для автоматизированного взаимодействия с веб-сервисами и электронной почтой на основе архитектуры ReAct и механизма суб-агентов. Достижение поставленной цели подразумевает создание полнофункциональной системы, способной автономно выполнять многошаговые сценарии, такие как фильтрация спама, извлечение данных из документов и навигация по динамическим веб-интерфейсам, с обеспечением строгого контроля безопасности.

Для достижения поставленной цели в работе были сформулированы и решены следующие **задачи**:

— Проведён анализ предметной области, эволюции систем автоматизации и существующих архитектурных паттернов интеллектуальных агентов

(ReAct, Tool Calling, Multi-Agent);

— Спроектирована многоуровневая архитектура агента, включающая ядро (AI Core), менеджер контекста, реестр инструментов и специализированный слой безопасности (Security Layer);
— Разработана подсистема изолированных суб-агентов для делегирования узкоспециализированных задач (анализ спама, извлечение сущностей из документов) с целью снижения уровня галлюцинаций базовой модели;
— Реализован программный комплекс на базе языка программирования Python с использованием фреймворка Playwright, протокола IMAP и API языковой модели GigaChat;
— Проведено тестирование системы на практических сценариях, включая автономную очистку почтового ящика и навигацию в веб-браузере; оценена эффективность предложенных механизмов безопасности и адаптации к ошибкам.

Объектом исследования выступают процессы автоматизации взаимодействия пользователя с веб-ресурсами и системами электронной почты. В рамках исследования рассматриваются как технические аспекты управления браузером и парсинга неструктурированных данных, так и вопросы обеспечения безопасности при автономном выполнении действий от имени пользователя.

Предметом исследования являются методы, алгоритмы и архитектурные решения для построения автономных интеллектуальных агентов с использованием механизмов Tool Calling, многоагентного взаимодействия и изолированных слоёв безопасности. Особое внимание уделяется вопросам минимизации рисков, связанных с галлюцинациями языковых моделей, и обеспечению устойчивости агента к изменениям пользовательских интерфейсов.

Научная новизна данной работы заключается в предложении гибридной архитектуры интеллектуального агента, объединяющей несколько ключевых компонентов:

— Внедрён выделенный слой безопасности (Security Layer), осуществляющий перехват и валидацию деструктивных команд (удаление, отправка, оплата) с запросом подтверждения у пользователя или применением строгих эвристик, что критически снижает риски автономной работы

LLM;

— Реализована архитектура суб-агентов (Sub-Agent Architecture), при которой сложные аналитические задачи, такие как классификация спама или анализ вложений, делегируются изолированным инстансам модели со специфичными системными промптами, что повышает точность анализа и предотвращает «загрязнение» основного контекста агента;
— Разработан механизм персистентных сессий (Persistent Sessions), позволяющий агенту бесшовно продолжать работу в авторизованных зонах веб-сервисов без необходимости повторного прохождения процедур аутентификации и обхода капчи;
— Предложен механизм семантического анализа DOM-дерева посредством инъекции JavaScript-кода, устраняющий зависимость от хрупких CSS/XPath-селекторов.

Практическая значимость работы заключается в возможности внедрения разработанного программного модуля для автоматизации офисных задач, первичной фильтрации входящей корреспонденции и сбора аналитической информации из открытых веб-источников. Решение не требует написания жёстких парсеров и способно адаптироваться к изменениям вёрстки сайтов. Система работает на стандартном компьютере, использует доступные API и может быть интегрирована в корпоративные рабочие процессы для снижения нагрузки на персонал.

В ходе выполнения работы использовался комплекс методов, охватывающих как теоретические, так и прикладные аспекты разработки. Методы анализа предметной области включали изучение современных паттернов проектирования ИИ-агентов и сравнительный анализ фреймворков автоматизации. Методы архитектурного проектирования применялись для построения модульной структуры с чётким разделением ответственности между AI-ядром, контроллером браузера и инструментами. Методы работы с веб-средой реализованы через библиотеку Playwright для извлечения снимков страницы (Page Snapshot) и взаимодействия с DOM-элементами по индексам. Методы работы с электронной почтой охватывают взаимодействие по протоколу IMAP, декодирование заголовков и извлечение вложений. Инструментальные средства включают язык программирования Python третьей версии, библиотеку

gigachat для интеграции с отечественной LLM, а также механизмы Prompt Engineering для обеспечения строгого JSON-формата ответов модели.

В первой главе бакалаврской работы проводится анализ предметной области и существующих решений. Рассмотрена эволюция систем автоматизации: от классических RPA-подходов (UiPath, Blue Prism, Selenium), страдающих от хрупкости и неспособности обрабатывать неструктурированные данные, к современным когнитивным системам на базе LLM. Проанализированы ключевые архитектурные паттерны интеллектуальных агентов:

— **Паттерн ReAct** (Reasoning and Acting) — объединение процессов рассуждения и выполнения действий в единый цикл «мысль–действие–наблюдение»;

— **Механизм Tool Calling** — предоставление модели реестра доступных функций с описанием сигнатур через JSON Schema для безопасного взаимодействия с внешним миром;

— **Архитектура Sub-Agent** — выделение узкоспециализированных изолированных агентов для решения разнородных задач без «загрязнения» основного контекста;

— **Слой безопасности** (Security Layer) — проактивная валидация деструктивных команд с парадигмой Human-in-the-Loop.

Проведён сравнительный анализ существующих фреймворков (AutoGPT, LangChain/LangGraph, Browser Use), выявлены их ограничения: отсутствие встроенного слоя безопасности, невозможность работы с протоколом IMAP, необходимость ручного проектирования графов состояний. На основании анализа сформулирована постановка проблемы: необходимость разработки архитектуры автономного агента, способного бесшовно переключаться между веб-интерфейсами и прямым взаимодействием с почтовыми серверами, обладающего встроенным слоем безопасности и использующего архитектуру суб-агентов для изоляции аналитических задач.

Во второй главе представлено проектирование архитектуры автономного интеллектуального агента. Архитектура системы разделена на пять независимых логических слоёв, взаимодействующих через строго определённые интерфейсы:

1. **Уровень оркестрации и управления** (Orchestration Layer) — цен-

тральный компонент (класс BrowserAgent), отвечающий за инициализацию подсистем, ведение диалогового цикла ReAct, маршрутизацию задач и управление состоянием сессии;

2. **Интеллектуальное ядро** (AI Core) — слой взаимодействия с крупной языковой моделью GigaChat, отвечающий за генерацию структурированных команд в формате JSON;
3. **Реестр и подсистема инструментов** (Tool Registry & Execution Layer) — набор изолированных модулей (BrowserTools, EmailTools) для навигации по DOM-дереву, отправки IMAP-запросов, извлечения данных;
4. **Слой безопасности** (Security Layer) — промежуточный фильтр, классифицирующий действия по уровню риска и инициирующий запрос подтверждения для деструктивных операций;
5. **Слой аналитических суб-агентов** (Sub-Agent Layer) — изолированные инстансы языковой модели со специализированными промптами для классификации спама и извлечения сущностей из документов.

Описан механизм управления контекстом (Context Manager), реализующий три уровня хранения данных: скользящее окно диалога (conversation_history), кэш промежуточных результатов (analysis_cache) и список посещенных страниц (visited_urls). Детально рассмотрена подсистема инструментов: контроллер веб-браузера на базе Playwright с семантическим анализом DOM через инъекцию JavaScript, подсистема работы с электронной почтой через протокол IMAP4, механизм персистентных сессий для обхода процедур многофакторной аутентификации.

В третьей главе описана программная реализация системы. Программный комплекс разработан на языке Python 3.10+ с использованием следующих ключевых технологий:

— **Playwright** (версия 1.40.0+) — фреймворк для автоматизации веб-браузеров с поддержкой асинхронного выполнения и механизма сохранения состояния сессии;
— **GigaChat SDK** (версия 0.1.15+) — официальный клиент для взаимодействия с отечественной большой языковой моделью, используемый в качестве основного интеллектуального ядра;
— **imaplib** — стандартная библиотека Python для работы с протоколом

IMAP4, обеспечивающая прямое и безопасное взаимодействие с почтовыми серверами;

— **python-dotenv** — инструмент для управления переменными окружения и конфиденциальными данными.

Реализован класс BrowserAgent, организующий итеративный цикл think_and_act с паттерном ReAct. Особенностью реализации является строгий контроль формата ответов LLM: системный промпт содержит явную инструкцию генерировать исключительно валидный JSON-объект, а утилита extract_json_from_text применяет алгоритм подсчёта скобок для надёжного парсинга. Внедрён механизм коррекции (Self-Correction), позволяющий агенту восстанавливаться после ошибок формата ответа модели.

Подсистема взаимодействия с веб-средой (класс BrowserTools) реализует семантический анализ DOM-дерева: инструмент extract_page_snapshot выполняет обход интерактивных элементов страницы и формирует их структурированное представление с уникальными индексами, что позволяет языковой модели оперировать не хрупкими селекторами, а семантическим содержимым. Реализован механизм обработки страниц безопасности и капчи: при обнаружении блокировки агент автоматически приостанавливает выполнение и передаёт управление человеку.

Подсистема работы с электронной почтой (класс EmailTools) обеспечивает извлечение писем за указанный период с автоматическим декодированием заголовков, безопасное удаление писем с обязательным подтверждением через Security Layer и генерацию текстовых отчётов. Архитектура суб-агентов (класс SubAgent) реализует специализированные инстансы для классификации спама (Spam Analyzer) и извлечения ключевых тезисов из документов (Document Analyzer).

В четвёртой главе проведена практическая демонстрация и тестирование системы. Запуск системы осуществляется через главный модуль main.py после настройки переменных окружения в файле .env и однократной подготовки персистентной сессии браузера. Протестированы следующие ключевые сценарии:

— **Автономная навигация в веб-браузере**: задача «Найди погоду в Москве» выполнена успешно за 3 шага за 11 секунд; агент коррект-

но сформировал URL, выполнил навигацию, извлёк и семантически проанализировал содержимое страницы результатов поиска Яндекса;

— **Классификация и удаление спама через IMAP**: из 4 полученных писем агент идентифицировал и удалил 2 письма с признаками спама; ошибка суб-агента была корректно обработана через резервный алгоритм эвристической классификации; слой безопасности пропустил команду удаления без дополнительного подтверждения, так как действие соответствовало исходной задаче пользователя;

— **Обработка капчи и страниц безопасности**: при обнаружении страницы с капчей агент автоматически приостановил выполнение, вывел уведомление и активировал инструмент wait_for_user; после ручного решения капчи пользователем работа продолжилась бесшовно;

— **Устойчивость к галлюцинациям LLM**: попытка модели вызвать несуществующий инструмент send_money была успешно заблокирована слоем валидации, после чего модель скорректировала план действий.

Результаты тестирования представлены в таблице:

Сценарий	**Успешность**	**Среднее время**	**Примечание**
Веб-навигация	95%	12 сек.	Требует обхода капчи
Чтение почты (IMAP)	100%	3 сек.	Стабильное соединение
Классификация спама	92%	8 сек.	Зависит от LLM
Удаление писем	100%	5 сек.	С подтверждением
Анализ документов	90%	8 сек.	Через суб-агент
Защита от галлюцинаций	100%	—	Блокировка 100%

Проведено сравнение разработанной системы с традиционными подходами: в отличие от RPA-систем, агент использует семантический анализ DOM и адаптируется к изменениям вёрстки без вмешательства разработчика; по сравнению с ручной обработкой почты время выполнения рутинных операций сократилось с 15–20 минут до 10–15 секунд на пакет из 10 писем; в отличие от изолированных чат-ботов, разработанный агент обладает полноценным доступом к инструментам операционной системы.

Заключение. В рамках данной бакалаврской работы решена задача проектирования и программной реализации автономного интеллектуально-

го агента для безопасного взаимодействия с веб-сервисами и электронной почтой. Исследование охватывает анализ паттернов многоагентных систем, механизмов Tool Calling и разработку полнофункционального прототипа на базе отечественной LLM GigaChat.

По итогам работы достигнуты следующие **результаты**:

— Проведён анализ предметной области: выявлены ограничения классических RPA-решений и пробелы в безопасности современных фреймворков, обосновавшие необходимость гибридной архитектуры;

— Спроектирована многоуровневая система из пяти слоёв: оркестрации, AI Core, реестра инструментов, Security Layer и подсистемы суб-агентов; паттерн ReAct со строгим JSON-форматом минимизировал ошибки парсинга и оптимизировал расход контекста;

— Реализована архитектура суб-агентов: делегирование аналитических задач изолированным инстансам LLM предотвратило «загрязнение» основного контекста и повысило точность обработки неструктурированных данных;

— Внедрён проактивный слой безопасности: механизм контекстной валидации деструктивных команд с парадигмой Human-in-the-Loop обеспечил баланс между автономностью агента и контролем со стороны оператора;

— Создан и протестирован программный комплекс на базе Python, Playwright, IMAP4 и GigaChat: тестирование подтвердило 100% защиту от несанкционированных действий и 92% точность классификации спама.

Практическая значимость заключается в готовности модуля к интеграции в корпоративные процессы для автоматизации фильтрации почты и сбора веб-данных без поддержки жёстких парсеров. Решение функционирует на стандартном оборудовании и обеспечивает прозрачный аудит всех автономных действий.

Перспективы развития включают:

— Разработку Telegram-интерфейса для мониторинга выполнения задач;

— Расширение реестра инструментов через REST API для интеграции с корпоративными системами;

— Внедрение RAG-механизмов для обеспечения долгосрочной памяти

агента;

— Переход к параллельной мультиагентной оркестрации для ускорения выполнения сложных сценариев.

Таким образом, поставленные во введении цель и задачи выполнены в полном объёме. Разработанный комплекс устойчив к изменениям интерфейсов и галлюцинациям LLM, готов к практическому применению и может служить методической базой для дальнейших исследований в области безопасных автономных систем.