МИНОБРНАУКИ РОССИИ
федеральное государственное автономное образовательное учреждение высшего образования
«Национальный исследовательский университет
«Московский институт электронной техники»

Институт системной и программной инженерии
и информационных технологий (СПИНТех)

Джугели Дмитрий Александрович

Выпускная квалификационная работа
по направлению 09.03.04 «Программная инженерия»

Разработка программного модуля для автоматизированного тестирования веб-форм.

Выполнил:
студент группы ПИН-51Д _____ /Д. А. Джугели/

Руководитель:
к. пед. н., доцент института СПИНТех _____ /Е. Л. Федотова/

Москва, 2026

СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ ИСПОЛЬЗУЕМЫХ СОКРАЩЕНИЙ

ПМ АТВ — программный модуль для автоматизированного тестирования веб-форм

SMTP — Simple Mail Transfer Protocol – протокол прикладного уровня, предназначенный для передачи электронной почты

API — Application Programming Interface — интерфейс программирования приложений

DOM — Document Object Model — независимый программный интерфейс, позволяющий получать доступ к HTML-документу и изменять его структуру

HTTP — HyperText Transfer Protocol — протокол прикладного уровня для передачи гипертекстовых документов

REST — Representational State Transfer — архитектурный стиль взаимодействия компонентов распределенного

WebSocket — протокол связи поверх TCP-соединения, предназначенный для обмена сообщениями между браузером и веб-сервером, используя постоянное соединение

WebSocket-сервер — серверная компонента, обеспечивающая двустороннюю связь в реальном времени между браузерным расширением и сервисом временной почты

Side Panel — боковая панель браузерного расширения

Popup — элемент интерфейса, появляющийся поверх основного окна

Парсинг — автоматизированный анализ и преобразование структурированных данных

JSON — текстовый формат обмена данными

UUID — универсальный уникальный идентификатор

ПО — программное обеспечение

ОС — операционная система

UI — пользовательский интерфейс

ВВЕДЕНИЕ

В современной веб-разработке и обеспечении качества программного обеспечения значительное количество времени занимает процесс тестирования, в частности, регрессионное и функциональное тестирование веб-форм. Такие формы часто требуют ввода уникальных данных, включая адреса электронной почты с последующей верификацией через письмо. Этот процесс является рутинным, отнимает значительное время у тестировщиков и разработчиков и подвержен человеческим ошибкам.

Актуальность разработки программного модуля для автоматизированного тестирования веб-форм обусловлена необходимостью сокращения временных затрат и повышения надежности тестирования за счет автоматизации генерации тестовых данных, использования временных почтовых адресов и взаимодействия с веб-интерфейсом в реальном времени.

Назначением разрабатываемого программного модуля (шифр ПМ АТВ) является автоматизация процесса тестирования веб-фор, требующих email-верификации, за счет создания интегрированного решения, состоящего из серверной части (SMTP-сервис) и клиентской части (браузерное расширение). Область применения модуля включает тестирование веб-приложений в условиях разработки и обеспечения качества программного обеспечения.

Целью выпускной квалификационной работы является разработка программного модуля для автоматизированного тестирования веб-форм, обеспечивающего сокращение временных трудозатрат на процесс тестирования.

Пояснительная записка состоит из перечня используемых сокращений, введения, исследовательского раздела, конструкторского раздела, испытательного раздела, заключения, списка литературы [1] и трех приложений [2, 3].

В исследовательском разделе приводится исследование предметной области и сравнительный анализ существующих решений проблемы. На основании анализа выявляется необходимость разработки ПМ АТВ. Также в этом разделе формулируются цель и задачи работы, приводятся описание концептуальной модели, входных и выходных данных ПМ АТВ и требования к алгоритмам работы модуля.

В конструкторском разделе осуществляется выбор языка программирования и среды программирования для реализации ПМ АТВ. Кроме того, описывается алгоритм

работы программного модуля и некоторых его подпрограмм в словесном и графическом виде.

В испытательном разделе описывается процесс отладки и тестирования ПМ АТВ. Анализируются методы и средства для тестирования программного модуля. Также разрабатываются тест-кейсы и приводятся результаты модульного и интеграционного тестирования. Для доказательства достижения цели ВКР проводятся сравнительные испытания.

Приложение 1 содержит техническое задание на разработку ПМ АТВ [4].

Приложение 2 содержит фрагменты исходного кода ПМ АТВ [5].

Приложение 3 содержит руководство оператора ПМ АТВ.

Объем пояснительной записки составляет 64 листа без учета приложений.

1. ИССЛЕДОВАТЕЛЬСКИЙ РАЗДЕЛ

1.1 Исследование предметной области

В настоящее время скорость и качества выпуска программного обеспечения являются критически важными факторами для успеха любого IT-проекта. Одной из наиболее трудоемких и повторяющихся задач в процессе разработки ПО является тестирование. В условиях конкурентной среды, где скорость разработки является ключевым фактором успеха, процессы тестирования требуют оптимизации и автоматизации.

По стандарту ISTQB, тестирование – это процесс, содержащий виды активности жизненного цикла, как динамические, так и статические, касающиеся планирования, подготовки и оценки программного продукта и связанных с этим результатов работ с целью определить, что они соответствуют описанным требованиям, показать, что они подходят для заявленных целей и для определения дефектов [6].

Согласно ГОСТ Р ИСО МЭК 12207-99, жизненный цикл программного обеспечения включает вспомогательные процессы, среди которых выделяются верификация, аттестация (валидация), совместный анализ и аудит [7].

Верификация представляет собой процесс, направленный на подтверждение того, что программное обеспечение полностью соответствует требованиям и условиям, зафиксированным в технической документации.

Аттестация (валидация) предназначена для оценки того, насколько программный продукт удовлетворяет потребностям и ожиданиям конечных пользователей, а также соответствует своему функциональному назначению.

Совместный анализ используется для оценки текущего состояния проекта и его промежуточных результатов, что позволяет своевременно корректировать ход работ.

Аудит проводится с целью проверки соответствия выполняемых работ установленным требованиям и планам проекта.

На рисунке 1.1 представлена общая схема процесса тестирования.

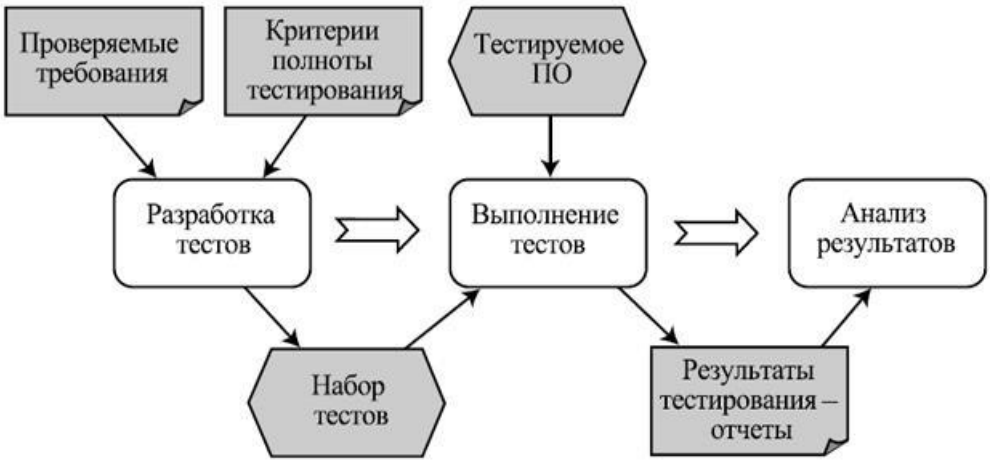

Рис. 1.1 Общая схема процесса тестирования

На начальном этапе инженер-тестировщик получает проверяемую программу и перечень требований к ее функционированию. Далее, опираясь на установленные критерии полноты тестирования, разрабатывается набор тестов, который должен охватить все ключевые аспекты работы программы. Набор тестов – комплект тестовых наборов для исследуемого компонента или системы, в котором обычно постусловие одного теста используется в качестве предусловия для последующего [6]. На следующем шаге происходит выполнение тестов, в ходе которого результаты тестирования фиксируются в отчетах. Эти данные анализируются для выявления соответствия или несоответствия программы заявленным требованиям. Тест включает в себя специально сформированную ситуацию и перечень наблюдений, которые необходимо провести для проверки программного обеспечения на соответствие заданным требованиям.

Согласно стандарту ГОСТ Р ИСО/МЭК 25010-2015, тестирование напрямую влияет на такие характеристики ПО, как функциональная пригодность, производительность, совместимость, удобство использования и защищенность [8].

По данным World Quality Report 2023, 35-40% бюджета IT-проектов уходит на тестирование, при этом дефекты, обнаруженные на поздних этапах разработки, обходятся в 5-10 раз дороже, чем выявление на ранних стадиях [9]. Согласно отчету Gartner (2022). Например, сбой в платежной системе интернет-магазина может привести к потере клиентов и репутационному ущербу. По данным Gartner, средняя стоимость простоя IT-системы составляет 5.600 долларов в минуту [10].

Особое место в процессе тестирования веб-приложений занимает проверка функциональности, связанной с пользовательским интерфейсом. Веб-формы являются

основным способом взаимодействия пользователя с серверной частью приложения. Их тестирование включает проверку:

– функциональной корректности (валидация полей, обработка спецсимволов и кириллицы, корректность работы капчи и антиспам-механизмов);

– удобство использования (интуитивность взаимодействия с элементами страницы, скорость отклика системы на действия пользователя);

– безопасности (защита от атак типа SQL-инъекций, XSS, CSRF);

– производительности (время обработки запроса на стороне сервера, нагрузка на систему при массовых запросах).

Общим для сценариев тестирования веб-форм является необходимость ввода и последующей проверки адреса электронной почты и других данных, что создает дополнительные сложности в процессе тестирования. Стандартный процесс тестирования веб-форм с email-подтверждением включает следующие этапы:

– ввод тестовых данных в поля формы;

– отправка формы и ожидание письма верификации;

– переключение в почтовый клиент;

– поиск и открытие письма от тестируемой системы;

– извлечение верификационной ссылки или кода из содержимого письма;

– возврат в тестируемое приложение и ввод полученных данных;

– проверка корректности завершения процесса.

Для целей тестирования можно выделить несколько ключевых типов веб-форм, каждый из которых характеризуется уникальными задачами и разным уровнем сложности. Простые контактные формы требуют минимальной валидации данных, и основной фокус при их проверке смещается на удобство пользователя (UX). Формы регистрации представляют собой более сложный объект, так как требуют проверки введенных данных на уникальность в системе. Наибольшую техническую сложность для тестирования представляют e-commerce формы, поскольку они подразумевают интеграцию с внешними платежными системами и шлюзами. Отдельной категорией являются административные панели, в которых формы становятся частью комплексной бизнес-логики и требуют тщательной проверки ролевого доступа и связных процессов.

Метрики сложности тестирования различных типов форм приведены в таблице 1.1

Таблица 1.1

Сравнительный анализ сложности тестирования веб-форм

Тип формы	Контактная форма	Форма регистрация	Форма оплаты	Многостраничный опрос
Количество тестовых сценариев	8 - 12	25 - 40	35 - 50	50 - 80
Среднее время ручного тестирования, минут	15 – 20	45 – 60	60 – 90	90 – 120
Критические точки	Валидация email, обработка спецсимволов	Email верификация, уникальность данных	Интеграция с платежными системами	Сохранение промежуточных данных

Временные затраты на одну итерацию такого тестирования составляют в среднем от 2 до 10 минут, в зависимости от скорости доставки почты. При необходимости выполнения десятков или сотен тестовых сценариев (включая различные комбинации тестовых данных) совокупные временные затраты становятся значительными.

Качественные проблемы ручного тестирования веб-форм включают:

– человеческий фактор – опечатки при вводе данных, использование уже зарегистрированных email-адресов, пропуск писем;

– воспроизводимость дефекта – сложность точного повторения сценария при регрессионном тестировании;

– синхронизация времени – задержки в доставки почте приводят к простою тестировщика;

– организационные сложности – необходимость иметь множество реальных или временных почтовых ящиков;

– ограничения бесплатных почтовых сервисов – лимиты на количество писем, блокировка за подозрительную активность.

Особенности тестирования для российского рынка включают необходимость использования реалистичных данных, которые в полной мере соответствуют региональным стандартам. Ключевыми аспектами являются проверка корректной обработки специфичных форматов данных, таких как структура ФИО, номера телефонов с актуальными кодами операторов и федеральными префиксами, а также почтовые адреса с правильными индексами и названиями городов России. Особое внимание уделяется поддержке кириллических символов в различных кодировках (UTF-8, Windows-1251), что критически важно для корректного отображения и обработки текста.

Экономический аспект проблемы заключается в том, что затраты на ручное тестирование растут пропорционально сложности системы. При этом автоматизация с использованием существующих инструментов, хоть и предлагает выход, при этом сопряжена со значительными издержками. Для ее успешной реализации необходимы квалифицированные специалисты, владеющие языками программирования. Кроме того, процесс часто усложняется необходимостью интеграции нескольких разнородных инструментов в единый контур, что требует дополнительных ресурсов.

Компании, инвестирующие в автоматизацию тестирования, сокращают время вывода продукта на рынок на 20-30% и уменьшают количество пост-релизных дефектов на 40-50% [10].

Разрабатываемый программный модуль призван решить эти проблемы путем создания интегрированного решения, состоящего из двух основных компонентов: сервиса временной электронной почты (SMTP-сервер) и браузерного расширения.

Практическая значимость данной выпускной квалификационной работы определяется потенциальным эффектом от внедрения разрабатываемого программного модуля для автоматизированного тестирования веб-форм.

Автоматизация процессов тестирования позволит существенно сократить временные затраты на подготовку и выполнение тестов. Ожидается, что внедрение модуля позволит уменьшить время, затрачиваемое на тестирование веб-форм на 70-80%.

Кроме того, использование ПМ АТВ позволит значительно уменьшить количество ошибок, связанных с ручным вводом данных. Это повысит точность и надежность тестирования, а также улучшит качество собираемых данных. Существенное сокращение ошибок положительно скажется на общей эффективности команды тестировщиков.

Таким образом, разработка специализированного программного модуля для автоматизированного тестирования веб-форм представляет собой актуальную задачу, решение которой позволит компаниям, занимающимся веб-разработкой, значительно сократить время на выполнение рутинных операций и повысить эффективность тестирования. Более подробно перспективы разработки ПМ АТВ представлены в таблице 1.2

Таблица 1.2

Перспективы разработки ПМ АТВ

До разработки ПМ АТВ	После разработки ПМ АТВ
Ручная генерация тестовых данных	Автоматическая генерация тестовых данных
Ручное заполнение веб-форм	Автоматическое заполнение веб-форм
Отсутствие временных email-адресов	Удобство использования временных email-адресов
Ручная обработка входящих писем	Автоматическая обработка входящих писем
Высокое количество ошибок при заполнении веб-форм	Существенное сокращение количества ошибок
Длительное время тестирования веб-форм	Сокращение времени тестирования на 70-80%

Разработка и внедрение ПМ АТВ позволит оптимизировать процесс тестирования веб-форм, повысить его эффективность и снизить нагрузку на специалистов, занимающихся тестированием.

1.2 Обзор существующих решений

В данной главе проведен сравнительный анализ существующих программных решений, обладающих функциональностью, схожей с разрабатываемым программным модулем для автоматизированного тестирования веб-форм (ПМ АТВ). Основное внимание уделено инструментам, которые так или иначе решают задачи генерации тестовых данных, работы с временной электронной почтой и автоматизацией взаимодействия с веб-формами.

Критерии выбора для сравнительного анализа сформулированы на основе требований к разрабатываемому модулю и включают:

– интеграцию в браузер – наличие браузерного расширения для непосредственного взаимодействия с веб-формами;

– генерацию тестовых данных – возможность создания реалистичных данных, специфичных для российского региона;

– автоподстановку в формы – автоматическое заполнение полей веб-формы сгенерированными данными;

– работу с временной почтой – создание временных email-адресов, прием писем;

– просмотр писем в реальном времени – мгновенное отображение входящих писем в интерфейсе инструмента;

– удобство использования – низкий порог входа, отсутствие необходимости в программировании;

– наличие API – возможность интеграции в автоматизированные процессы тестирования;

– стоимость – доступность бесплатного использования;

– локализацию – поддержку русского языка в интерфейсе.

Анализируемые решения разделены на четыре категории в зависимости от их основной функциональности: сервисы временной электронной почты, инструменты автоматизации тестирования, библиотеки генерации тестовых данных и браузерные расширения.

Сервисы временной электронной почты:

Guerrilla Mail – один из старейших сервисов временной почты, предоставляющий одноразовые email-адреса. Сервис работает через веб-интерфейс, не требует регистрации и поддерживает прием писем в течение 60 минут. Основное преимущество – бесплатный доступ к веб-сервису и простота использования. Однако Guerrilla Mail не предоставляет браузерное расширение для автозаполнения форм, не поддерживает генерацию тестовых данных и не имеет полноценного API для интеграции в автоматизированные процессы.

Temp-Mail – популярный сервис временной электронной почты с широким набором доменов для генерации адресов. В отличие от Guerrilla Mail, Temp-Mail предоставляет REST API для программного взаимодействия, что позволяет автоматизировать создание почтовых ящиков и получение писем. Однако бесплатная версия имеет ограничение в 100 запросов в сутки и не предоставляет доступ к содержимому писем в текстовом формате. Не имеет браузерного расширения для автозаполнения для автозаполнения форм и не решает задачу генерации тестовых данных.

Mailtrap – профессиональный сервис для тестирования email в разработке. Предоставляет виртуальный SMTP-сервер, который перехватывает все отправляемые письма и отображает их в веб-интерфейсе без реальной отправки получателям. Однако данный сервис не решает задачи автозаполнения веб-форм и не имеет браузерного расширения.

Инструменты автоматизации тестирования:

Selenium WebDriver – промышленный стандарт для автоматизации тестирования веб-приложений. Представляет собой мощный фреймворк, позволяющий эмулировать любые действия пользователя в браузере, включая заполнение форм, клики по элементам и навигацию по страницам. Selenium не предоставляет готового пользовательского интерфейса для взаимодействия с email и автозаполнением форм.

Библиотеки генерации тестовых данных:

Faker – мощная библиотека для генерации реалистичных тестовых данных. Поддерживает множество локалей, включая русскую, что позволяет создавать аутентичные ФИО, адреса, номера телефонов и другие данные, специфичные для России. Однако библиотека является компонентом, а не полноценным решением. Она не предоставляет временные email-адреса и не имеет механизма автозаполнения веб-форм.

Браузерные расширения для тестирования:

Fake Data Generator – браузерное расширение, которое позволяет генерировать тестовые данные (имена, адреса, номера кредитных карт) и вставлять их в поля форм. Данное расширение не имеет функционала для взаимодействия с email-адресами.

Результаты сравнения существующих аналогичных решений по перечисленным выше критериям приведены в таблице 1.3

Таблица 1.3

Сравнительный анализ существующих аналогов

Программное решение Критерий	Guerrilla Mail[11]	Temp Mail[12]	Selenium[13] + Faker[14]	Mailtrap[15]	ПМ АТВ
Интеграция в браузер (расширение)	-	-	+-	-	+
Генерация данных (российский регион)	-	-	+	-	+
Автоподстановка в формы	-	-	+	-	+
Наличие временной почты	+	+	-	+	+
Просмотр писем в реальном времени	+-	+-	-	+	+
Стоимость подписки (руб/мес)	785	800	0	1200	0
Локализация (интерфейс на русском языке)	-	+	-	-	+

Обозначения:

«+» – функция присутствует;

«-» – функция отсутствует;

«+-» - функция присутствует частично или с существенными ограничениями.

Как видно из таблицы, ни одно из аналогичных решений не удовлетворяет всем требованиям в рамках единого, простого в использовании инструмента. Задача для создания программного модуля ПМ АТВ является актуальной.

Разрабатываемый модуль будет представлять собой законченное решение, ориентированное на тестировщиков и разработчиков, позволяющее выполнять сложные сценарии тестирования с минимальными временными затратами. Он сократит время тестирования одного сценария с email-верификацией с 5-10 минут до 1 минуты, уменьшит вероятность ошибок, связанных с человеческим фактором, и повысит эффективность работы тестировщиков за счёт автоматизации рутинных операций. ПМ АТВ обеспечит воспроизводимость тестовых сценариев для регрессионного тестирования и предоставит удобный интерфейс, не требующий навыков программирования.

1.3 Цель и задачи разработки ПМ АТВ

На основе анализа, проведенного в пунктах 1.1 и 1.2, определены цели и задачи разработки ПМ АТВ.

Цель: сокращение временных затрат и трудозатрат на процесс тестирования веб-форм, требующих электронной верификации и заполнения данных.

Для достижения поставленной цели были поставлены и выполнены следующие задачи: исследование предметной области, включающее анализ проблематики тестирования веб-форм с email-верификацией и изучения существующих подходов и инструментов; проведение сравнительного анализа существующих программных решений с оценкой аналогов по ключевым критериям и выявлением их преимуществ и недостатков; формулировка требований к разрабатываемому модулю на основе проведенного анализа предметной области; разработка концептуальной модели и проектирование архитектуры модуля; разработка пользовательского интерфейса в виде браузерного расширения; программная реализация ПМ АТВ; тестирования модуля, охватывающее юнит-тесты, интеграционное и функциональное тестирование, а также разработка сопроводительной документации.

1.4 Описание концептуальной модели

Разрабатываемый программный модуль для автоматизированного тестирования веб-форм (ПМ АТВ) представляет собой распределенную систему, состоящую из двух основных компонентов: серверной части (SMTP-сервис с REST API) клиентской части

(браузерное расширение). Система реализует сквозную автоматизацию процесса тестирования веб-форм с использованием email-верификации. На рисунке 1.2 представлена общая архитектура ПМ АТВ.

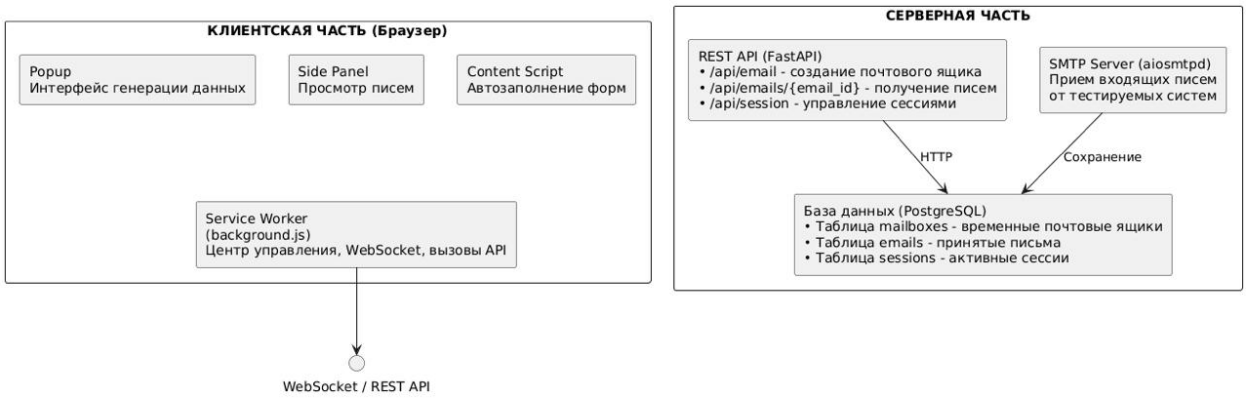

Рис. 1.2 Общая архитектура ПМ АТВ

Варианты использования включают:

– генерацию тестовых данных – создание реалистичный данных для российского региона (ФИО, адрес, телефон);

– создание временного почтового ящика – генерация уникального email-адреса для получения писем;

– автозаполнение веб-формы – автоматическая подстановка данных в поля формы;

– просмотр входящих писем – отображение полученных писем в браузерном расширении.

Концептуальная модель программного модуля представляет собой архитектурную структуру, описывающую основные компоненты системы, их взаимодействие и потоки данных. Эта модель служит основой для разработки, обеспечивая четкое понимание функциональности и взаимосвязей между различными частями системы. Она позволяет лучше понять, как работает модуль тестирования, и как его компоненты взаимодействуют друг с другом для достижения конечного результата – автоматизации процесса тестирования веб-форм с email-верификацией.

Пользователь (тестировщик) взаимодействует с модулем через интерфейс браузерного расширения, который является ключевым элементом системы. Интерфейс

включает в себя всплывающее окно (Popup) для генерации данных и управления процессом тестирования, а также боковую панель (Side Panel) для просмотра входящих писем в реальном времени. Пользовательские действия, а именно: нажатие кнопок «Сгенерировать данные» и «Сгенерировать почту», генерируют события, которые затем обрабатываются системой для выполнения соответствующих действий.

Любое действие пользователя в рамках модуля отслеживается системой слушателей событий. Эти инструменты делятся на несколько категорий: слушатели браузерного API, слушатели расширения и обработчики WebSocket-соединений. Слушатели браузерного API отлавливают события, генерируемые пользователем: клики по элементам интерфейса, навигацию по страницам и взаимодействие с веб-формами. События расширения обрабатываются сервис-воркером, который инициирует соответствующие действия в модуле, такие как вызов REST API сервера или установка WebSocket-соединения.

Ключевым элементом системы является менеджер состояния, который отвечает за хранение и управление текущим состоянием модуля. Объект состояния содержит информацию о сгенерированных тестовых данных, активных временных почтовых ящиках, текущей сессии WebSocket и параметрах автозаполнения форм. Менеджер состояния предоставляет методы для обновления состояния на основе пользовательских действий и входящих данных от сервера. Эти методы позволяют синхронизировать данные между различными компонентами расширения и обеспечивать согласованность интерфейса.

Менеджер автозаполнения (Content Script Manager) отвечает за взаимодействие с DOM тестируемых веб-страниц. Он включает в себя методы для поиска полей формы, определения их типов и автоматической подстановки сгенерированных данных. Эти методы обеспечивают корректное заполнение веб-форм без вмешательства пользователя, используя CSS-селекторы и анализ атрибутов HTML-элементов. Менеджер так же предотвращает конфликты с нативным поведением браузера и обеспечивает безопасность при взаимодействии с контентом страниц.

Сервис-воркер является центральным компонентом клиентской части, управляющим жизненным циклом расширения. Он отвечает за установку и подержание WebSocket-соединения с сервером, обработку входящих уведомлений о новых письмах и выполнении асинхронных вызовов REST API. Сервис-воркер обеспечивает фоновую

работу модуля даже при закрытом popup-интерфейсе, что позволяет получать уведомления в реальном времени.

Серверная часть модуля построена на асинхронной архитектуре и включает несколько основных компонентов. SMTP-сервер на базе aiosmtpd постоянно ожидает входящие соединения, принимает письма от тестируемых приложений и передает их на обработку. Парсер электронной почты извлекает структурированные данные из сообщений, преобразуя сырые данные в формат, пригодный для хранения и отображения. WebSocket-сервер управляет подключениями клиентов и обеспечивает рассылку уведомлений о новых событиях.

REST API предоставляет интерфейс для управления почтовыми ящиками и получения данных.

База данных (PostgreSQL) выступает в роли хранилища состояния серверной части, сохраняя информацию о временных почтовых ящиках, принятых письмах и активных сессиях. Основные таблицы: mailboxes - временные почтовые ящики, emails – принятые письма, sessions – активные сессии.

Таким образом, концептуальная модель ПМ АТВ представляет собой сложную, но четко структурированную систему, где каждый компонент выполняет специфическую функцию, а их взаимодействие обеспечивает сквозную автоматизацию процесса тестирования веб-форм.

Взаимодействие компонентов системы осуществляется по следующей схеме:

1. Клиент-серверное взаимодействие через REST API для управления почтовыми ящиками;
2. Двусторонняя связь через WebSocket для уведомлений о новых письмах в реальном времени;
3. Прием почты через SMTP для получения писем от тестируемых систем;
4. Взаимодействие с DOM через Content Scripts для автозаполнения веб-форм.

На рисунке 1.3. представлена диаграмма последовательности для основного сценария работы системы – тестирования веб-формы с email-верификацией.

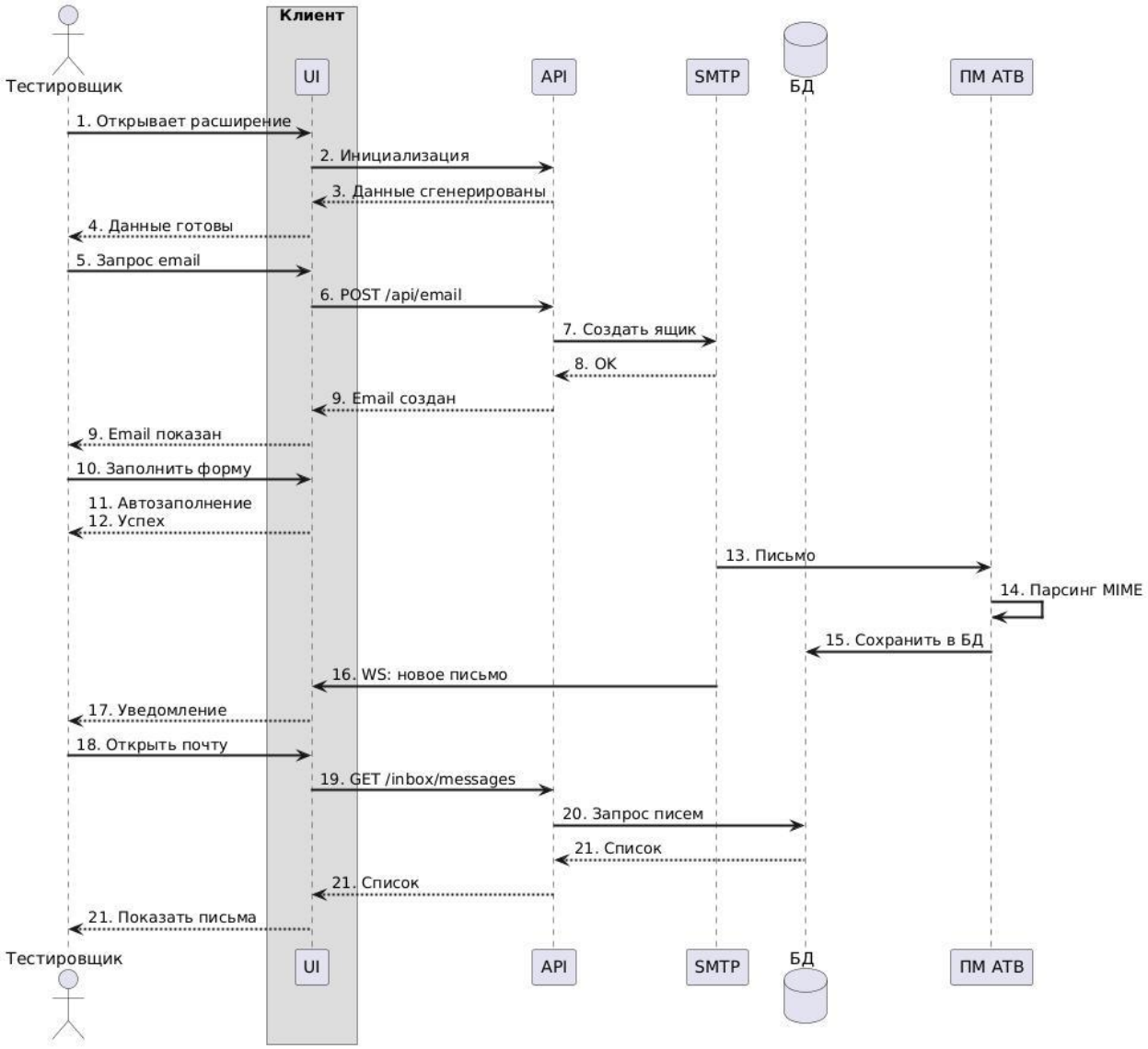

Рис. 1.3 Диаграмма последовательности тестирования веб-формы с помощью ПМ АТВ

Последовательность действий: Пользователь открывает popup-интерфейс расширения и нажимает на кнопку «Сгенерировать данные», Расширение генерирует тестовые данные, Пользователь нажимает кнопку «Сгенерировать почту», Расширение отправляет POST-запрос на /api/email к серверу, Сервер создает временный почтовый ящик, сохраняет его в БД и возвращает адрес, Content Script находит поля формы и заполняет их сгенерированными данными, Пользователь отправляет форму в тестируемом приложении, Тестируемое приложение отправляет письмо на временный адрес, SMTP-сервер принимает письмо, парсит его и сохраняет в БД, WebSocket-сервер от-

правляет уведомление всем подключенным клиентам, Расширение получает уведомление и обновляет список писем, Пользователь просматривает полученное письмо, не выходя из текущей вкладки.

1.5 Входные и выходные данные

На основании концептуальной модели модуля автоматизированного тестирования веб-форм, определены наборы входных и выходных данных и форматы их внутреннего представления в программе. Организация данных направлена на обеспечения эффективности обработки и соответствия требованиям процесса тестирования.

Входные данными для программы являются:

– действия пользователя в браузерном расширении (нажатие кнопок "Сгенерировать данные" и "Сгенерировать почту" в Popup-интерфейсе);

– внешние входящие email-сообщения (письма, отправленные тестируемым веб-приложением на сгенерированные временные адреса);

– веб-страницы, с которыми взаимодействует пользователь.

Выходными данными, отправляемыми разрабатываемым модулем мобильному оборудованию, являются:

– сгенерированные тестовые данные (массивы объектов в формате JSON);

– временные адреса электронной почты;

– структурированная информация о входящих письмах.

Для эффективного управления входными и выходными данными в системе реализована специализированная схема хранения информации, которая обеспечивает целостность и согласованность данных на всех этапах работы модуля. Данная схема оптимизирована для работы в режиме реального времени и поддерживает быстрый обмен информацией между клиентской и серверной частями системы.

1.6 Требования к разрабатываемому ПМ АТВ

Модуль автоматизированного тестирования веб-форм должен обеспечивать выполнение следующих функциональных и нефункциональных требований.

Функциональные требования:

– система должна генерировать реалистичные тестовые данные для российского региона;

– система должна управлять временной электронной почтой: создание временных email-адресов, прием входящих писем, автоматическое удаление почтовых ящиков;

– система должна автоматически определять поля веб-формы и подставлять сгенерированные данные;

– система должна отображать список входящих сообщений на временный почтовый адрес.

Нефункциональные требования:

– время генерации тестовых данных < 500 мс;

– время создания временного почтового ящика < 400 мс;

– интуитивно понятный пользовательский интерфейс;

– система должна сократить время тестирования одного сценария с 5-10 минут до 1 минуты;

– система должна снизить количество ошибок при ручном тестировании.

Выводы по разделу

В данном разделе проведено исследование предметной области и выявлена ее проблематика. Обозначены проблемы существующих ручных и полуавтоматизированных подходов при тестировании веб-форм, такие как высокие временные затраты и вероятность ошибок. Далее проведен аналитический обзор существующих решений и инструментов. Анализ показал, что на рынке отсутствует единое решение, сочетающее в себе простоту браузерного расширения, генератор тестовых данных, функциональность сервиса временной почты и интерактивность в реальном времени. Разработана концептуальная модель программного модуля. Для концептуальной модели определен формат входных и выходных данных. Также сформулированы требования к разрабатываемому программному модулю.

2. КОНСТРУКТОРСКИЙ РАЗДЕЛ

2.1 Выбор языка программирования для серверной части ПМ АТВ

Разработка программного модуля для автоматизированного тестирования веб-форм (ПМ АТВ) требует тщательного выбора языка программирования для серверной части, так как от этого зависит производительность, надежность и скорость разработки системы. Серверная часть ПМ АТВ включает в себя SMTP-сервис для приема и обработки электронных писем, REST API для управления почтовыми ящиками и WebSocket-сервер для уведомлений в реальном времени. Таким образов, язык программирования должен обеспечивать высокую производительность, поддержку асинхронных операций, наличие библиотек для работы с почтовыми протоколами и удобство разработки API.

В настоящее время существует более 700 официально зарегистрированных языков программирования, из которых необходимо выбрать тот, которые наиболее подходит для разработки серверной части ПМ АТВ.

Для выбора языка программирования были сформулированы ключевые критерии. Во-первых, важными аспектами являются производительность и поддержка асинхронной обработки запросов, поскольку сервер должен эффективно работать со множеством одновременных SMTP-соединений, HTTP-запросов и WebSocket-подключений. Во-вторых, язык должен обладать развитой экосистемой, включающей специализированные библиотеки для работы с SMTP, парсинга MIME-сообщений и обработки электронной почты. В-третьих, инструментарий должен обеспечивать высокую скорость разработки REST API, позволяя быстро создавать структурированные эндпоинты. В-четвертых, обязательным требованием является нативная или хорошо интегрированная поддержка протокола WebSocket для реализации обновлений в реальном времени.

Рассмотрены такие языки, как Java, C#, Go, Node.js, и Python, с учетом их особенностей, преимуществ и недостатков.

Java – высокоуровневый объектно-ориентированный язык программирования со статической типизацией, который широко применяется для разработки корпоративных приложений, веб-сервисов, мобильных приложений под системой Android, а так же

встраиваемых систем. Одним из ключевых преимуществ Java является его кроссплатформенность, обеспечиваемая виртуальной машиной Java (JVM), которая позволяет запускать Java-код на любой платформе без необходимости внесения изменений. Java известен своей высокой производительностью и надежностью, что делает его популярным для создания сложных и высоконагруженных систем. Язык обладает богатой экосистемой библиотек и фреймворков, таких как Spring Boot для разработки REST API и JavaMail для работы с электронной почтой. Java также поддерживает многопоточность, что позволяет эффективно обрабатывать множество одновременных запросов, и предоставляет мощные механизмы безопасности. Опыт разработки 6 месяцев.

Однако Java обладает и рядом недостатков. Код на Java может быть более громоздким и многословным по сравнению с другими языками, такими как Python, что замедляет процесс разработки. Строгая типизация и необходимость писать больше кода для реализации даже простых задач могут увеличивать время на разработку. Кроме того, настройка асинхронной обработки в Java может быть более сложной по сравнению с языками, изначально ориентированными на асинхронное программирование, такими как Node.js или Python с asyncio.

C# - современный объектно-ориентированный язык программирования со строгой статической типизацией, разработанный компанией Microsoft. Он широко используется для создания корпоративных приложений, веб-сервисов, десктоп приложений и игра благодаря своей мощной интеграцией с платформой .NET. C# сочетает в себе простоту синтаксиса с высокой производительностью, что делает его популярным выбором для разработки сложных и масштабируемых систем. Опыт разработки 1 год.

Тем не менее, у C# есть и некоторые ограничения. Экосистема библиотек для работы с электронной почтой и парсингом MIME-сообщений менее развита по сравнению с Python или Node.js, что может потребовать дополнительных усилий для интеграции необходимых функций. Разработка на C# также требует использования платформы .NET, что может ограничивать выбор операционных систем и сред выполнения.

Golang – компилируемый язык программирования, разработанный компанией Google, который отличается простотой, высокой производительностью и встроенной поддержкой параллелизма. Go компилируется в машинный код, что обеспечивает высокую скорость выполнения программ. Одним из ключевых преимуществ Go является

его встроенная поддержка параллелизма через горутины и каналы, что позволяет эффективно обрабатывать множество задач одновременно.

Однако у Go есть некоторые недостатки. Хотя его экосистема для веб-разработки развивается, она все еще уступает по разнообразию экосистемам таких языков, как Python и JavaScript. Это может ограничивать выбор инструментов и библиотек для решения специфичных задач. Опыт разработки отсутствует.

Node.js – это среда выполнения JavaScript, которая позволяет запускать JavaScript- код на сервере. Node.js известен своей асинхронной, событийно-ориентированной архитектурой, что позволяет эффективно обрабатывать множество одновременных соединений. Одним из главных преимуществ Node.js является его богатая экосистема библиотек и модулей, доступных через npm.

К недостаткам Node.js можно отнести динамическую типизацию, которая может приводить к ошибкам, которые сложно обнаружить на этапе разработки. Кроме того, Node.js не так хорошо подходит для задач, требующих интенсивных вычислений, из-за однопоточной природы. Опыт разработки отсутствует.

Python – интерпретируемый язык программирования с динамической типизацией, который известен своим простым и читаемым синтаксисом. Это делает его популярным выбором для быстрой разработки и прототипирования, особенно в таких областях, как веб-разработка, машинное обучение и анализ данных. Python обладает богатой экосистемой библиотек, включая такие фреймворки, как Django и Flask для веб-разработки, а также библиотеки для работы с электронной почтой, такие как aiosmtpd. Поддержка асинхронного программирования с использованием библиотек, таких как asyncio, позволяет Python эффективно обрабатывать асинхронные задачи, такие как работа с сетевыми запросами и WebSocket. Большое и активное сообщество Python облегчает поиск решений и поддержку проектов. Опыт разработки 1 год.

Тем не менее, Python имеет и некоторые недостатки. Будучи интерпретируемым языком, он может уступать в производительности компилируемым языкам, таким как Go или Java, особенно в задачах, требующих высокой вычислительной мощности.

Навыки и опыт работы с конкретным языком программирования могут значительно повлиять на эффективность разработки. Глубокое знание языка позволяет разработчику минимизировать количество ошибок и оперативно их исправлять, что существенно ускоряет процесс создания программного обеспечения.

Сравнение рассматриваемых языков программирования по перечисленным критериям приведено в таблице 2.1.

Таблица 2.1

Сравнение языков программирования для серверной части

Язык / Характеристики	Java [16]	C# [17]	Go [18]	Node.js [19]	Python[20]
Производительность, запросы в секунду	~2500	~3000	~3500	~2000	~2500
Поддержка асинхронности	+- CompletableFuture	+ async/await	+ goroutine	+ event loop	+ asyncio
Библиотеки для SMTP/Email	+- JavaMail	+ MailKit	+ Stdlib	+ Nodemailer	+ Aiosmtpd
Поддержка WebSocket	+- Spring WS	+ SignalR	+ Gorilla WS	+ Socket.IO	+ Websockets
Популярность языка (место в рейтинге IEEE Spectrum)	2	7	8	Нет данных	1
Опыт разработки на языке, лет	0,5	1	0	0	1

Условные обозначения:

«+» - присутствует;

«+-» - присутствует частично;

«-» - отсутствует.

По результатам сравнения получается, что наиболее подходящим языком для написания серверной части программного модуля для автоматизированного тестирования веб-форм является Python. Он выделяется простотой и скоростью разработки, а также богатой экосистемой библиотек, что делает его оптимальным для создания надежного REST API и работы с SMTP.

2.2 Выбор языка программирования для клиентской части ПМ АТВ

Разработка клиентской части программного модуля для автоматизированного тестирования веб-форм (ПМ АТВ) предполагает создание браузерного расширения, которое должно обеспечивать генерацию тестовых данных, автозаполнение веб-форм и взаимодействие с серверной частью через WebSocket. Клиентская часть должна быть совместима с браузерными API, а также обеспечивать удобство разработки и надежность кода.

Выбор языка программирования для клиентской части является критически важным, так как он определяет не только функциональность и производительность расширения, но и удобство его разработки и поддержки. В данном разделе проведен анализ языков программирования, которые могут быть использованы для реализации клиентской части ПМ АТВ, с учетом следующих критериев:

– Совместимость с браузерными API. Язык должен иметь полный доступ к DOM, браузерным событиям и API расширений.

– Интеграция с UI-фреймворками. Возможность использования современных фреймворков для построения интерфейса Popup и Side Panel.

– Опыт работы. Преимущество будет у языков, опыт разработки на которых больше, что ускорит реализацию проекта.

Рассмотрены следующие языки: JavaScript, TypeScript, Dart, WebAssembly.

JavaScript – это основной язык программирования для создания клиентской части веб-приложений. Он поддерживается всеми современными браузерами, что делает его кроссплатформенным решением для разработки браузерных решений. JavaScript позволяет создавать динамические веб-страницы и имеет встроенную поддержку браузерного API, что обеспечивает доступ к DOM-дереву, обработку событий и взаимодействие с пользовательским интерфейсом. Одним из ключевых преимуществ JavaScript является наличие большого количества библиотек и фреймворков. Опыт разработки 6 месяцев.

Тем не менее, JavaScript имеет и некоторые недостатки. Одним из основных минусов является динамическая типизация, которая может приводить к труднонаходимым ошибкам. По мере увеличения объема кода его поддержка и рефакторинг становятся более сложными, что может замедлить процесс разработки.

TypeScript – это язык программирования, разработанный компанией Microsoft на основе JavaScript. Основным преимуществом данного языка является добавление строгой типизации, что позволяет выявлять ошибки на этапе компиляции, а не выполнения. Это значительно повышает надежность кода и упрощает его поддержку. TypeScript компилируется в JavaScript, что обеспечивает его совместимость со всеми браузерами и платформами, где выполняется JavaScript. Опыт разработки 3 месяца.

Одним из недостатков TypeScript является увеличение времени разработки по сравнению с JavaScript из-за необходимости прописывать типы данных.

Dart – язык программирования, разработанный компанией Google, который компилируется в JavaScript. Dart известен своей строгой типизацией и высокой производительностью, что делает его подходящим для разработки веб-приложений.

Тем не менее, Dart имеет ограниченную экосистему для разработки браузерных расширений. Сообщество разработчиков Dart значительно меньше, чем у TypeScript, что может затруднить поиск решений и поддержку проектов. Опыт разработки отсутствует.

WebAssembly – это технология, позволяющая запускать код, написанный на языках C++ или Rust в браузере. WebAssembly предоставляет возможность выполнения высокопроизводительных вычислений непосредственно в браузере.

Однако WebAssembly не имеет прямого доступа к DOM и браузерным API. Опыт разработки отсутствует.

Сравнение рассматриваемых языков программирования по перечисленным критериям приведено в таблице 2.2.

Таблица 2.2

Сравнение языков программирования для клиентской части

Язык Характеристики	TypeScript [21]	Dart [22]	WebAssembly [23]	JavaScript [24]
Совместимость с браузерами	Компилируется в JS	Требует компиляции	Требует JS-прослойки	Нативный
Прямой доступ к DOM и API браузера	Полный	Ограниченный (через js-interop)	Отсутствует (только через JS)	Полный
Популярность языка (место в рейтинге IEEE Spectrum)	5	46	19	3
Врем компиляции/сборки, сек	1-3	3-10	30+	0
Размер runtime, КБ	0	50 КБ	100 КБ	0
Опыт разработки на языке, лет	0,25	0	0	0,5

Таким образом, для разработки клиентской части ПМ АТВ наиболее подходящими языками программирования являются JavaScript и TypeScript. Несмотря на то, что TypeScript предлагает строгую типизацию, что повышает надежность кода, его использование влечет за собой дополнительные этапы компиляции и усложняет конфигурацию среды разработки. Для разработки клиентской части ПМ АТВ наиболее подходящим языком является JavaScript.

2.3 Выбор среды разработки

Разработка программного модуля для автоматизированного тестирования вебформ (ПМ АТВ) включает в себя работу над двумя основными компонентами: серверной частью, реализованной на языке Python, и браузерным расширением, написанным на JavaScript. Выбор среды разработки является важным этапом, так как он определяет эффективность работы, скорость разработки и удобство поддержки кода. Среда разработки должна обеспечивать глубокую интеграцию с выбранными языками программирования, поддерживать необходимые инструменты для работы с базами данных, фреймворками и системами контроля версий.

В данном разделе проведен анализ сред разработки, которые могут быть использованы для реализации ПМ АТВ. Рассмотрены следующие среды разработки: PyCharm, WebStorm, Visual Studio Code, IntelliJ IDEA.

PyCharm – это специализированная интегрированная среда разработки от компании JetBrains, ориентированная на разработку на языке Python. PyCharm предоставляет превосходную поддержку Python, включая автодополнение кода, рефакторинг и отладку. Среда поддерживает работу с фреймворками, такими как FastAPI и Django.

Тем не менее, PyCharm имеет и некоторые недостатки. Поддержка JavaScript в PyCharm обеспечивается через плагины, но не является нативной. Кроме того, PyCharm требует платной лицензии для полноценного использования.

WebStorm – это специализированная интегрированная среда разработки от JetBrains, ориентированная на веб-разработку с использованием TypeScript. Данная среда разработки обеспечивает отличную поддержку TypeScript, включая автодополнение, рефакторинг и отладку. Среда предоставляет удобные инструменты для работы с фреймворками React, Angular, Vue.js.

Однако поддержка Python в WebStorm ограничена и требует установки дополнительных плагинов.

IntelliJ IDEA – универсальная интегрированная среда разработки от компании JetBrains, поддерживающая множество языков программирования через плагины. IntelliJ IDEA обеспечивает поддержку как Python, так и JavaScript, но уступает специализированным IDE в глубине интеграции.

Тем не менее, IntelliJ IDEA требует платной лицензии, что является недостатком.

Visual Studio Code – бесплатный кроссплатформенный редактор кода от компании Microsoft, который обладает огромной библиотекой расширений. Visual Studio Code обеспечивает высокую степень поддержки как для Python, так и для JavaScript благодаря расширениям. Среда предоставляет удобные инструменты для отладки, рефакторинга, управления версиями кода, что делает ее подходящей для работы над проектами любой сложности. Среда полностью бесплатна и имеет богатую экосистему расширений.

Во время разработки программы необходимо отслеживать вносимые в ее текст изменения, чтобы всегда иметь возможность вернуться к последней рабочей версии кода. Для этого существуют различные системы контроля версий, среди которых одной

из наиболее популярных является git. В Visual Studio Code реализован удобный инструмент для работы с git непосредственно в интерфейсе среды разработки.

Опыт работы с некоторой IDE является преимуществом при выборе. Также для платных сред на приобретение лицензии уходит некоторое время. По данному критерию для разработки ПМ АТВ подходит Visual Studio Code.

Сравнение рассматриваемых сред разработки по перечисленным критериям приведено в таблице 2.3.

Таблица 2.3

Сравнительный анализ сред разработки

Среда разработки / Критерий	PyCharm [25]	WebStorm [26]	IntelliJ IDEA [27]	Visual Studio Code [28]
Поддержка Python	+	-	+	+
Поддержка JavaScript	+-	+	+	+
Интеграция с Git	+	+	+	+
Количество доступных плагинов, штук	~3000	~2000	~5000	~40000
Производительность, секунд на запуск	~5-10	~5-10	~10-15	~1-5
Стоимость, рублей в год	~5000	~4500	~6000	Бесплатно

Условные обозначения:

«+» - присутствует;

«+-» - присутствует частично;

«-» - отсутствует.

Таким образом, для разработки ПМ АТВ наиболее подходящей средой разработки является Visual Studio Code. Она является единственным бесплатным инструментом, который обеспечивает поддержку обоих ключевых технологий проекта – Python и JavaScript, а также обладает самой богатой экосистемой расширений и высокой производительностью. Visual Studio Code позволяет удобно работать с обоими компонентами проекта, обеспечивая необходимую функциональность для эффективной разработки.

2.4 Алгоритм работы ПМ АТВ

Программный модуль для автоматизированного тестирования веб-форм (ПМ АТВ) начинает свою работу с момента активации пользователем браузерного расширения. Основной задачей модуля является автоматизация процесса тестирования веб-форм, включая генерацию тестовых данных, автозаполнение форм и обработку входящих электронных писем. ПМ АТВ состоит из двух основных компонентов: клиентской части, реализованной в виде браузерного расширения, и серверной части, включающей SMTP-сервер, REST API и WebSocket-сервер.

Клиентская часть ПМ АТВ взаимодействует с пользователем через интерфейс браузерного расширения, а также автоматически заполняет веб-формы тестовыми данными. Серверная часть отвечает за обработку почтовых сообщений, отправленных на временные email-адреса, и отправку уведомлений в реальном времени через WebSocket. Взаимодействие между клиентской и серверной частями осуществляется посредством HTTPS-запросов и WebSocket-соединений, что позволяет реализовать сквозной процесс тестирования веб-форм с email-верификацией.

Для ПМ АТВ выбрана клиент-серверная архитектура с WebSocket для коммуникации в реальном времени, схема архитектуры представлена на рисунке 2.1

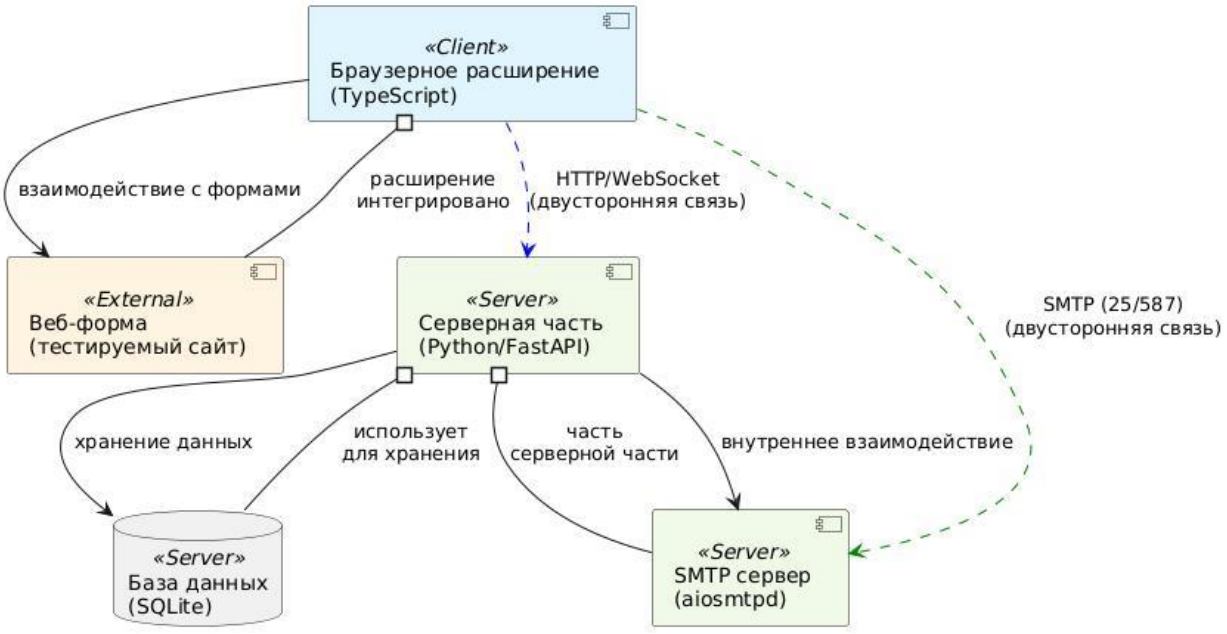

Рис. 2.1 Схема архитектуры ПМ АТВ

Для наглядности работы системы можно представить ее в виде двух взаимосвязанных алгоритмов: алгоритма работы клиентской части и алгоритма работы серверной части.

Клиентская часть ПМ АТВ реализуется в виде браузерного расширения, которое состоит из нескольких модулей: Popup-интерфейса, Content Script и Side Panel. Каждый из этих модулей выполняет определенные функции, обеспечивая взаимодействие с пользователем и автоматизацию процесса тестирования веб-форм.

Инициализация Popup-интерфейса. Пользователь открывает Popup-интерфейс браузерного расширения, в интерфейсе доступны кнопки для генерации тестовых данных и временного email-адреса.

Генерация тестовых данных. При нажатии на кнопку «Сгенерировать данные» расширение использует библиотеку Faker.js для создания массива реалистичных тестовых данных, таких как ФИО, адрес, телефон и другие необходимые поля. Эти данные отображаются в интерфейсе Popup для визуального контроля пользователем.

Запрос временного email-адреса. После генерации тестовых данных пользователь может нажать кнопку «Сгенерировать почту», что приводит к отправке асинхронного POST-запроса на серверную часть по эндпоинту /api/email. Серверная часть генерирует уникальный временный email-адрес, сохраняет его в базе данных и возвращает в ответе клиенту. Полученный email-адрес отображается в интерфейсе Popup.

Автозаполнение веб-формы. После получения временного email-адреса Content Script расширения, работающий в контексте активной веб-страницы, автоматически находит поля для ввода тестируемой формы и заполняет их сгенерированными данными. Это позволяет пользователю быстро и удобно протестировать веб-формы без необходимости ручного ввода данных.

Подписка на уведомления через WebSocket. Расширение устанавливает устойчивое WebSocket-соединение с сервером и подписывается на события, связанные с созданным почтовым ящиком. Это позволяет получать уведомления о новых письмах в реальном времени и оперативно обновлять интерфейс Side Panel.

Обработка входящих писем. При получении уведомления о новом письме через WebSocket расширение обновляет интерфейс Side Panel, добавляя письмо в список. Пользователь может просмотреть содержимое письма, не покидая текущую вкладку

браузера, что делает процесс тестирования удобным и эффективным. Схема алгоритма работы клиентской части представлена на рисунке 2.2

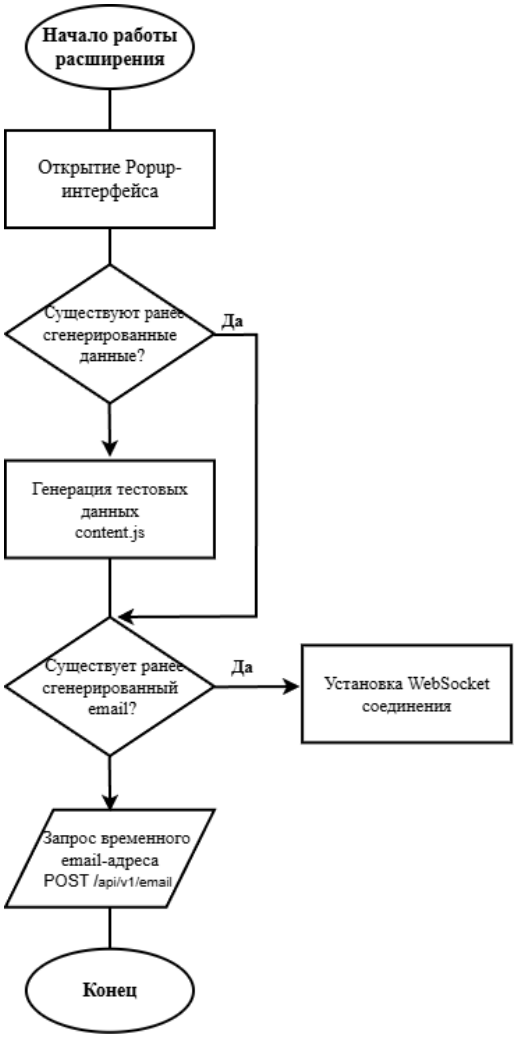

Рис. 2.2 Схема алгоритма клиентской части

Серверная часть ПМ АТВ реализуется на языке Python с использованием фреймворка FastAPI и включает в себя несколько ключевых модулей: SMTP-сервер, модуль парсинга писем, REST API и WebSocket-сервер.

Обработка запроса на генерацию временного email-адреса. Серверная часть принимает POST-запрос на эндпоинт /api/email, который инициируется клиентской частью для генерации временного email-адреса. Сервер генерирует уникальный email-адрес, сохраняет его в базе данных и возвращает в ответе клиенту.

Прием входящих писем через SMTP-сервер. SMTP-сервер, реализованный на базе библиотеке aiosmtpd, постоянно ожидает входящие почтовые сообщения. Когда

тестируемое веб-приложение отправляет письмо на сгенерированный адрес, SMTP-сервер принимает его и передает на обработку.

Парсинг и сохранение писем. Сервер парсит MIME-сообщения, извлекая ключевые данные, такие как отправитель, тема письма, текстовое и HTML-содержимое, а также вложения. Структурированные данные сохраняются в базе данных и связываются с соответствующим почтовым ящиком.

Рассылка уведомлений через WebSocket. При сохранении нового письма сервер инициирует рассылку уведомлений через WebSocket-сервер всем подключенным клиентам, которые подписаны на данный почтовый ящик. Это позволяет клиентской части оперативно получать информацию о новых письмах и обновлять интерфейс Side Panel. Схема алгоритма работы серверной части представлена на рисунке 2.3

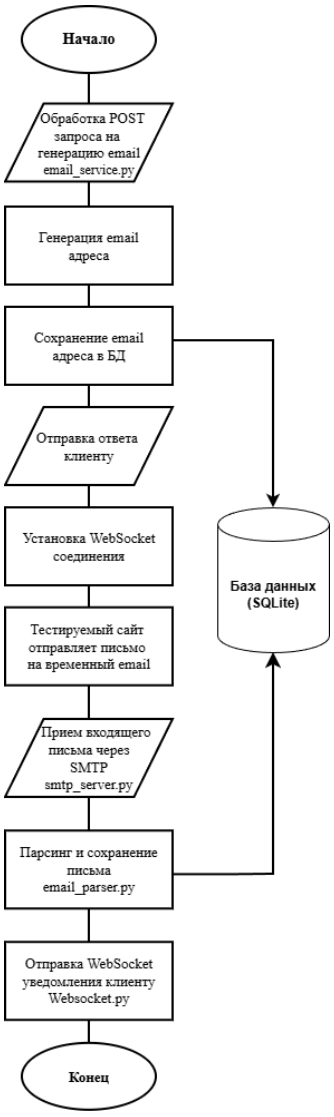

Рис. 2.3 Схема алгоритма серверной части

Взаимодействие клиентской и серверной частей. Взаимодействие между клиентской и серверной частями ПМ АТВ осуществляется посредством HTTP-запросов и WebSocket-соединений. Клиентская часть отправляется HTTP-запросы на сервер для генерации временных email-адресов и получения списка писем, а серверная часть обрабатывает эти запросы и отправляет уведомления о новых письмах через WebSocket.

Схема алгоритма ПМ АТВ представлена на рисунке 2.4

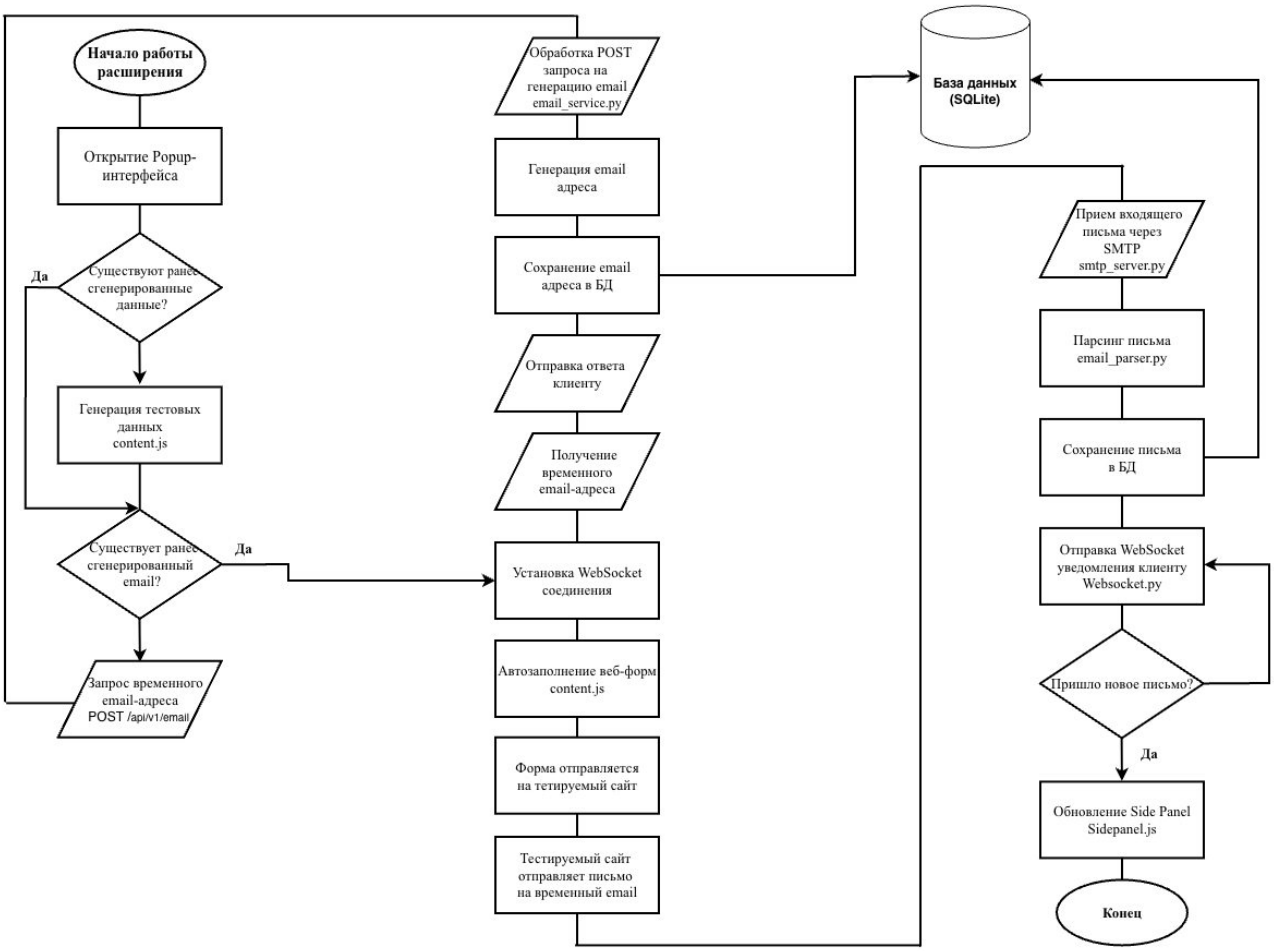

Рис. 2.4 Схема алгоритма ПМ АТВ

– начало работы - пользователь открывает Popup-интерфейс браузерного расширения, инициализируются обработчики событий для кнопок «Сгенерировать данные» и «Сгенерировать почту»;

– пользователь инициирует создание данных и почтового ящика через Popup;

– background-воркер выполняет API-вызовы и устанавливает WebSocket-соединение;

– content_script автоматически заполняет веб-форму;

– при отправке формы тестируемым приложением, серверная часть принимает письмо через smpt_server, парсит его через mail_parser и сохраняет в БД;

– модуль websocket рассылает уведомление на клиент;

– background-воркер расширения получает уведомление и делегирует обновление данных в side_panel, где пользователь видит письмо в реальном времени.

Таким образом, алгоритм работы ПМ АТВ представляет собой сложную систему взаимодействия клиентской и серверной частей, обеспечивающую автоматизацию процесса тестирования веб-форм с email-верификацией. Использование WebSocket позволяет реализовать уведомление в реальном времени, что делает процесс тестирования более удобным и эффективным.

2.5 Описание пользовательского интерфейса ПМ АТВ

Пользовательский интерфейс программного модуля для автоматизированного тестирования веб-форм (ПМ АТВ) является ключевым компонентом, обеспечивающим удобное и эффективное взаимодействие пользователя с системой. Основной задачей интерфейса является предоставление пользователю возможности генерации тестовых данных, автозаполнение веб-форм и просмотра входящих писем в реальном времени.

При разработке пользовательского интерфейса были учтены основные принципы проектирования интерфейсов, такие как минимализм, консистентность, отзывчивость и доступность.

Минимализм и простота являются основополагающими принципами проектирования интерфейса. Интерфейс лишен избыточных элементов, что позволяет пользователю сосредоточиться на основных задачах тестирования веб-форм. Все элементы управления, такие как кнопки генерации данных, поля для отображения сгенерированных данных и почтового адреса, а также интерфейс для просмотра входящих писем, расположены таким образом, чтобы быть легко доступными, но не отвлекать внимание от рабочего процесса.

Отзывчивость интерфейса достигается за счет оптимизации работы JavaScript-кода. Интерфейс мгновенно реагирует на действия пользователя, такие как нажатие на кнопки генерации данных или почтового адреса, а также обновление списка входящих писем.

Popup-интерфейс является основным элементом взаимодействия пользователя с ПМ АТВ. Он содержит кнопки для генерации тестовых данных и временного email-адреса, а также поля для отображения сгенерированных данных. Интерфейс выполнен в минималистичном стиле, что позволяет пользователю быстро ориентироваться и выполнять необходимые действия. На рисунке 2.5 представлена экранная форма Popup-интерфейса ПМ АТВ.

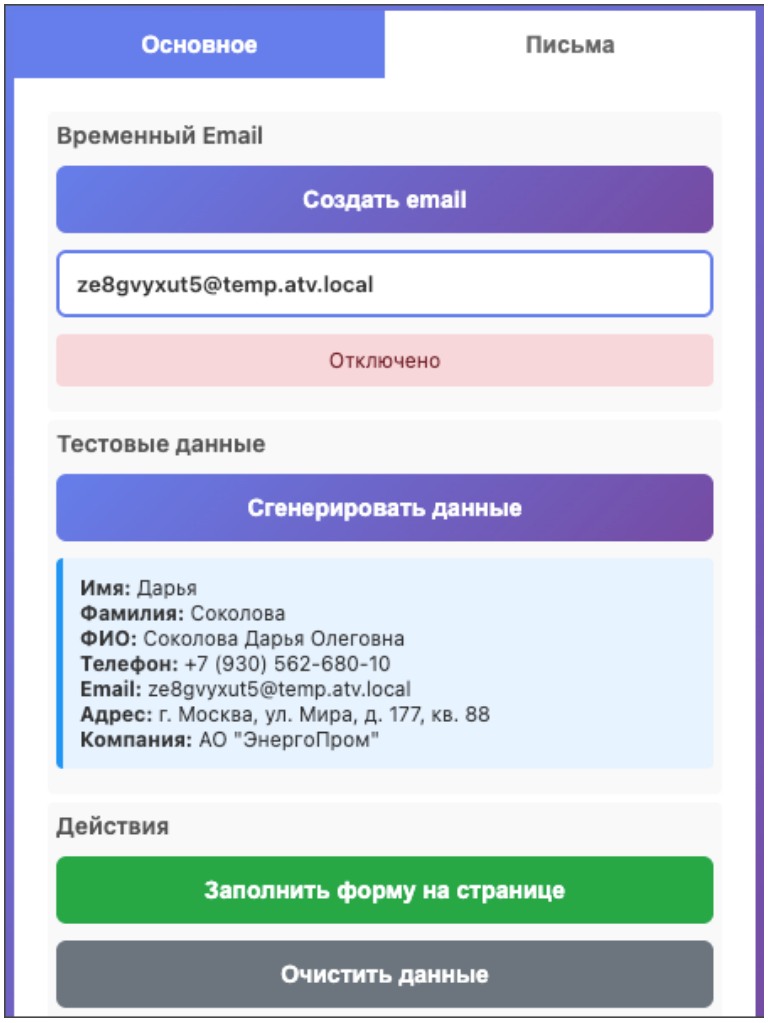

Рис. 2.5 Экранная форма Popup-интерфейса ПМ АТВ

Side Panel используется для отображения списка входящих писем и их содержимого. Пользователь может посмотреть тему письма, отправителя и содержимое письма, не покидая текущую вкладку браузера. Side Panel обновляется в реальном времени при получении новых писем через WebSocket-соединение. На рисунке 2.6 представлена экранная форма Side Panel ПМ АТВ.

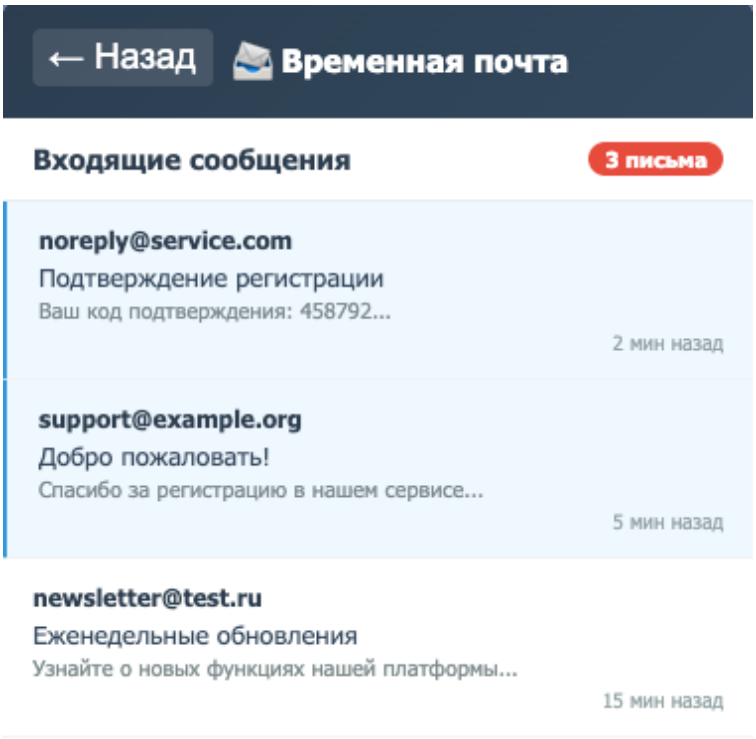

Рис. 2.6 Экранная форма Side Panel ПМ АТВ

Пользовательский интерфейс ПМ АТВ разработан с учетом современных требований к удобству, доступности и функциональности. Он обеспечивает пользователю удобное и эффективное взаимодействие с системой, позволяя быстро и легко выполнять задачи по тестированию веб-форм.

Выводы по разделу

В конструкторском разделе была проведена комплексная проработка архитектуры и алгоритмов работы программного модуля для автоматизированного тестирования веб-форм (ПМ АТВ). Был проведен анализ возможных языков программирования и сред разработки, с помощью которых решается поставленная задача. По результатам сравнения в качестве языка программирования серверной части выбран Python, для клиентской части JavaScript. В качестве среды разработки — Visual Studio Code, которая обеспечивает поддержку как Python, так и JavaScript. Также в этом разделе разработаны и описаны алгоритмы работы клиентской и серверной частей и составлены схемы алгоритма работы ПМ АТВ. Был разработан пользовательский интерфейс браузерного расширения, который обеспечивает удобное и эффективное взаимодействие пользователя с системой.

Таким образом, в конструкторском разделе были решены ключевые задачи, необходимые для успешной реализации ПМ АТВ.

3. ИСПЫТАТЕЛЬНЫЙ РАЗДЕЛ

3.1 Отладка ПМ АТВ

Важной частью разработки программного обеспечения является процесс поиска и устранения ошибок. Этот процесс можно разделить на два этапа: отладка и тестирование. Отладкой называется процесс поиска, локализации и устранения ошибок в программном коде. Цель отладки – обеспечить корректную, предсказуемую и стабильную работу программного обеспечения.

Процесс отладки программного модуля для автоматизированного тестирования веб-форм (ПМ АТВ) включает в себя поиск и устранение ошибок как на этапе разработки, так и на этапе тестирования. Отладка проводилась с использованием современных инструментов и сред разработки, таких как Visual Studio Code. Данная среда разработки обладает широкими возможностями для отладки различных типов приложений. VS Code имеет встроенную поддержку отладки JavaScript, для отладки кода на Python доступны различные расширения.

Отладка клиентской части, реализованной в виде браузерного расширения, проводилась с использованием Visual Studio Code и Chrome DevTools. Основное внимание уделялось проверке корректности работы скриптов, отвечающих за генерацию тестовых данных, взаимодействие с сервером и автозаполнение веб-форм.

Для отладки скрипта popup.js, отвечающего за генерацию тестовых данных и временных email-адресов, использовались точки останова в Visual Studio Code. Были проверены функции генерации данных, отправки запросов на сервер и обработки ответов. Например, при генерации тестовых данных с использованием библиотеки Faker.js отладчик помогал выявлять ошибки в логике работы функций, а также проверять корректность отправки запросов на сервер для генерации временных email-адресов.

Процесс отладки клиентской части в VS Code приведен на рисунке 3.1.

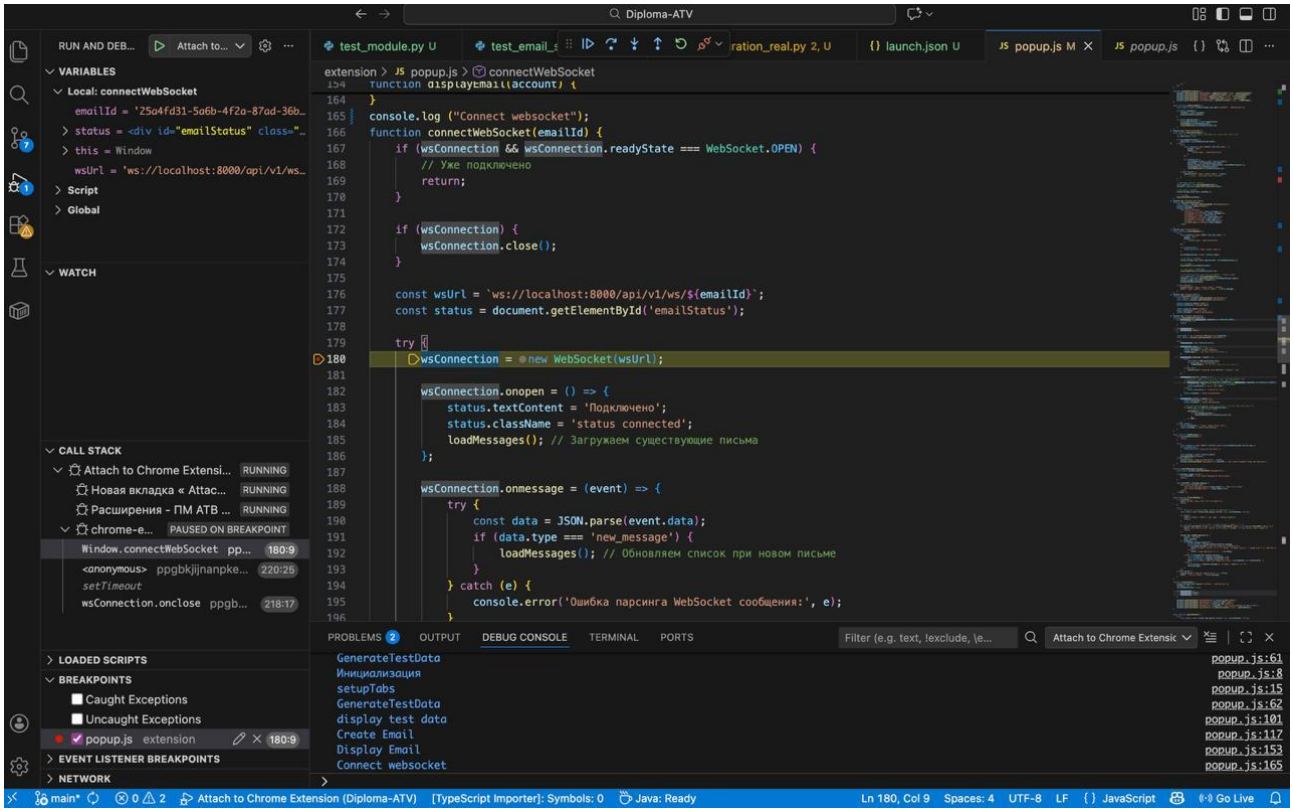

Рис. 3.1 Процесс отладки клиентской части в Visual Studio Code

Основным инструментом для отладки веб-приложений, включая браузерные расширения, является DevTools – панель разработчика, встроенная в современные браузеры. Одним из ключевых преимуществ этого инструмента является кроссбраузерность, то есть он доступен во всех популярных браузерах и вызывается одинаково – при помощи нажатия на клавишу F12.

Панель разработчика DevTools содержит множество вкладок с различными инструментами, которые активно применяются при отладке веб-приложений. Рассмотрим основные из них:

Вкладка Elements (Элементы) позволяет просматривать и редактировать HTML-разметку страницы. Этот инструмент используется на протяжении всего процесса разработки и отладки ПМ АТВ. Вкладка Elements играет особую роль при проверке корректности отображения элементов интерфейса, таких как Popup и Side Panel, а также при анализе структуры DOM-дерева. С ее помощью можно проверять корректность связей между элементами, корректность применения стилей, а также соответствие структуры HTML требованиям доступности.

Вкладка Console (Консоль) – это многофункциональный инструмент, который включает в себя несколько функциональных зон: панель вывода сообщений, командную строку и фильтр сообщений. Вывод сообщений консоль осуществляется с помощью объекта console и его методов, таких как log, error и warn. Особенно полезными являются методы time и timeEnd, которые позволяют замерять время выполнения отдельных функций или операций. Например, с помощью этих методов можно измерить время выполнения функции генерации тестовых данных.

Вкладка Performance (Производительность) предоставляет инструменты для анализа загрузки и выполнения кода, применяется для оптимизации выполнения JavaScript-кода, а также диагностики проблем с памятью.

Таким образом, использование инструментов DevTools позволяет эффективно выявлять и устранять ошибки, оптимизировать производительность и обеспечивать корректную работу ПМ АТВ.

Процесс отладки клиентской части в Chrome DevTools приведен на рисунке 3.2.

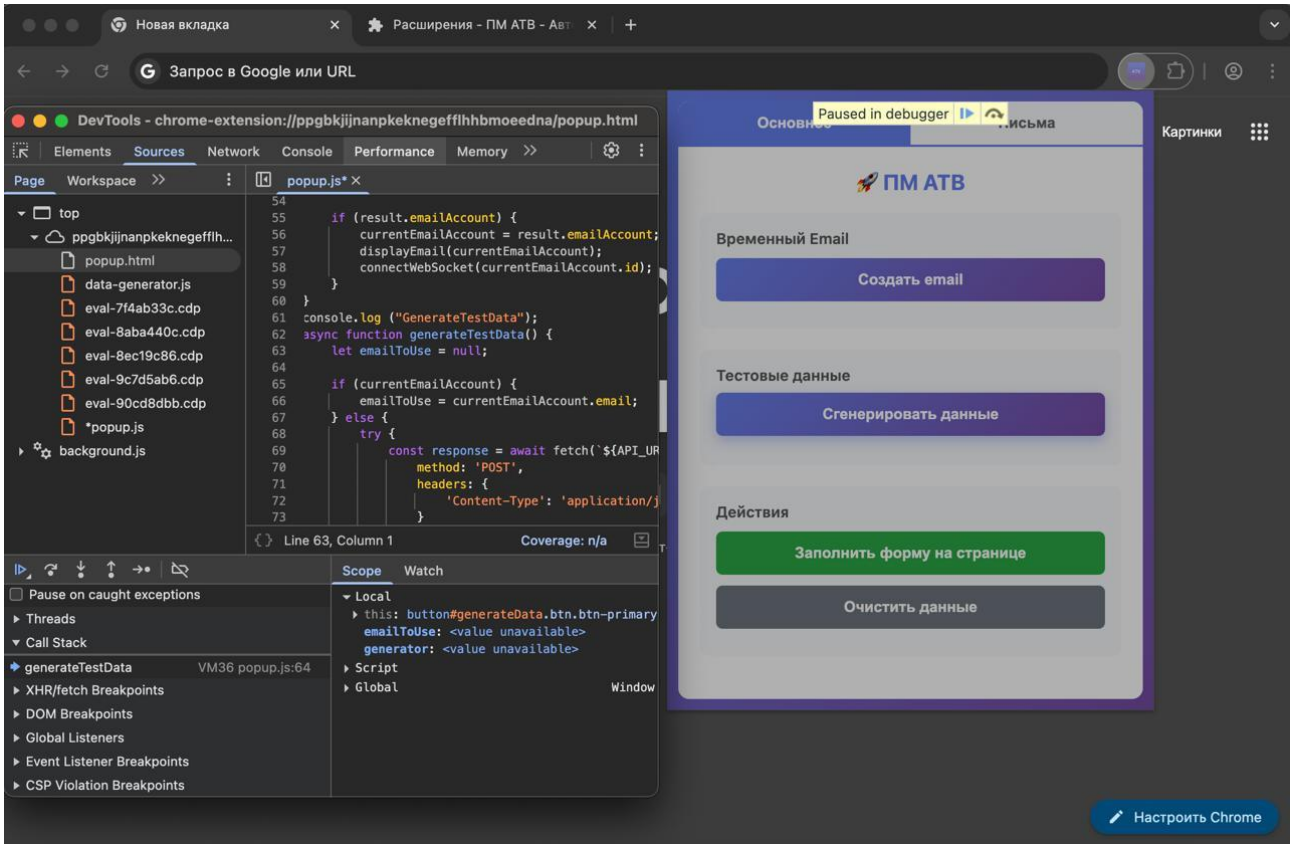

Рис. 3.2 Процесс отладки клиентской части в Chrome DevTools

Отладка серверной части, реализованной на Python с использованием фреймворка FastApi, включала проверку корректности обработки запросов, работы с базой данных и отправки уведомлений через WebSocket. Были установлены точки останова в ключевых местах кода. Например, при некорректной обработке входящих писем через SMTP-сервер отладчик помогал выявлять и исправлять ошибки в логике парсинга писем и взаимодействия с базой данных.

Процесс отладки серверной части в VS Code приведен на рисунке 3.3.

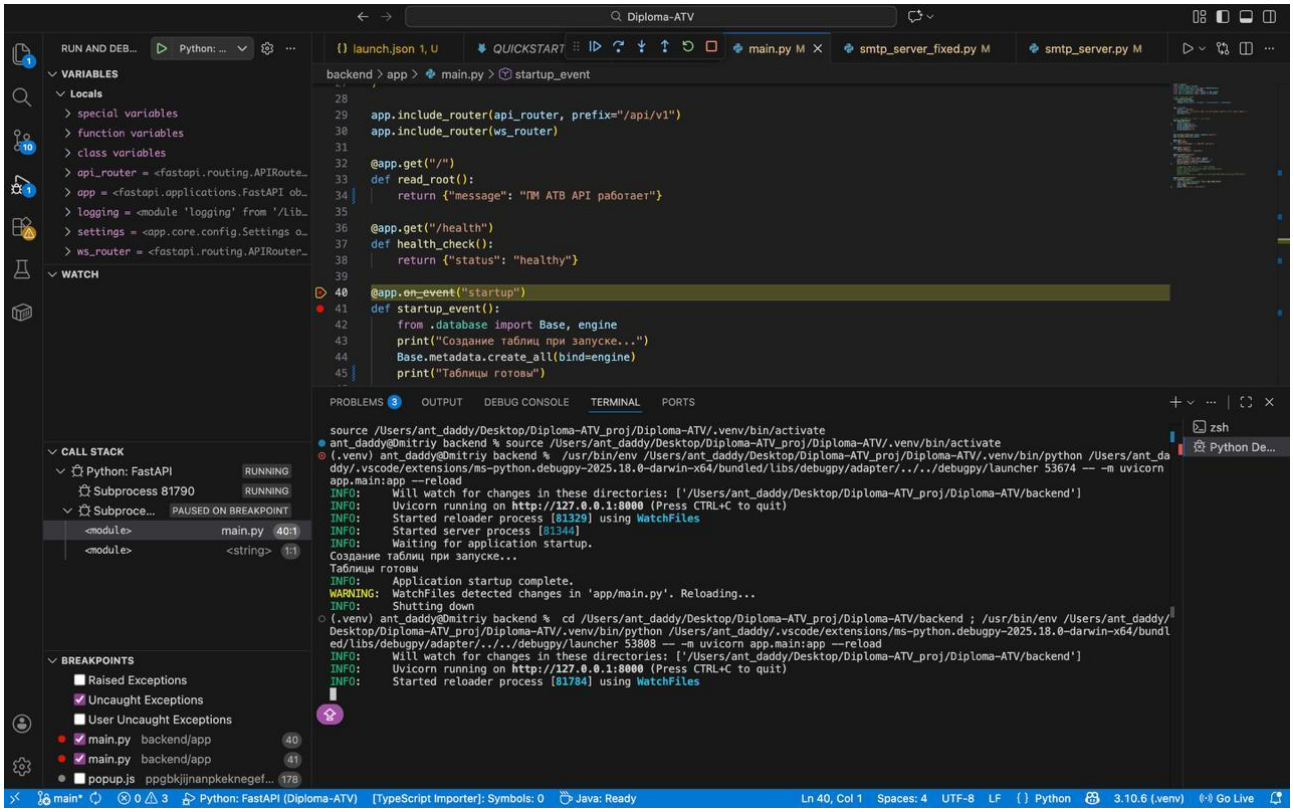

Рис. 3.3 Процесс отладки серверной части в VS Code

Таким образом, процесс отладки как клиентской, так и серверной части позволил выявить и устранить основные ошибки при разработке программного модуля ПМ АТВ.

3.2 Анализ методов тестирования ПМ АТВ

Тестирование программного обеспечения представляет собой комплексный процесс исследования разрабатываемого продукта на соответствие установленным требованиям. Для минимизации ошибок и недоработок, которые могут возникнуть на этапе

эксплуатации, используется многоуровневая система проверки качества. Этот подход, также известный как пирамида тестирования, включает в себя несколько этапов, каждый из которых направлен на проверку различных аспектов функционирования системы. Пирамида тестирования включает следующие этапы:

1. Модульное тестирование – начальный этап, направленный на проверку отдельных компонентов системы. Модульное тестирование позволяет выявлять ошибки на ранних этапах разработки, когда их исправление требует минимальных затрат.

2. Интеграционное тестирование – процесс, включающий проверку взаимодействия между различными модулями системы. На этом этапе проверяется корректность взаимодействия клиентской и серверной частей, а также их взаимодействие с внешними системами.

3. Системное тестирование – уровень тестирования, при котором приложение рассматривается как единое целое. Проверяется его соответствие техническим требованиям и бизнес-правилам. На этом этапе проверяется работа всего модуля в целом, включая взаимодействие с пользователем и внешними системами.

4. Приемочное тестирование – завершающий этап, на котором подтверждается готовность системы к промышленному использованию и соответствие всем установленным требованиям.

Схема пирамиды тестирования приведена на рисунке 3.4.

Рис. 3.4 Схема пирамиды тестирования

В контексте тестирования клиентской части веб-приложений важно отметить визуальные тесты и юзабилити тесты.

Визуальные тесты – тесты, которые включают в себя все проверки интерфейса веб-приложения. Они сравнивают визуальную часть приложения на разных версиях. Визуальные тесты нужны для обнаружения неожиданных изменений в UI.

Юзабилити тесты – это тесты, функционал которых фокусируется на удобстве работы с интерфейсом веб-приложения с точки зрения конечного пользователя.

Тестирование разделяется на два вида: ручное и автоматизированное.

Ручное тестирование используется для анализа сложных пользовательских сценариев, оценки удобства интерфейсов и исследования неочевидных кейсов. Например, при работе с REST API широко используются инструменты Postman и Swagger, которые предоставляют возможности для формирования HTTP-запросов и настройки параметров аутентификации. Ручное тестирование позволяет более гибко подходить к проверке функциональности, особенно в случаях, когда автоматизация затруднена.

Автоматизированное тестирование использует разработанные тестовые сценарии для проверки корректности программы. Оно требует написания тестов, что требует

большей компетенции тестировщика, но позволяет существенно сэкономить время при регрессионном тестировании.

Каждая ошибка в разной степени влияет на качество конечного продукта. Для структуризации и создания приоритетов дефекты классифицируются по степени влияния на работоспособность системы. Выделяются следующие уровни критичности:

– Блокирующие – дефекты, делающие систему непригодной для использования;

– Критические – ошибки в главных функциональных модулях;

– Значительные – частичная потеря функциональности смежных модулей;

– Незначительные – дефекты, не влияющие на работоспособность системы.

Важным аспектом тестирования является уровень доступа к внутренней структуре системы, что определяет выбор подхода к проверке ее функциональности. Существует три основных стратегии, каждая из которых имеет свои особенности и область применения.

Тестирование методом черного ящика – проверка внешнего поведения системы на основе документации, без знания внутренней структуры проекта.

Тестирования методом белого ящика – исследование внутренней логики и алгоритмов системы.

Тестирование методом серого ящика – комбинированный подход, сочетающий метод черного и белого ящика.

Различия методов тестирования представлены на рисунке 3.5.

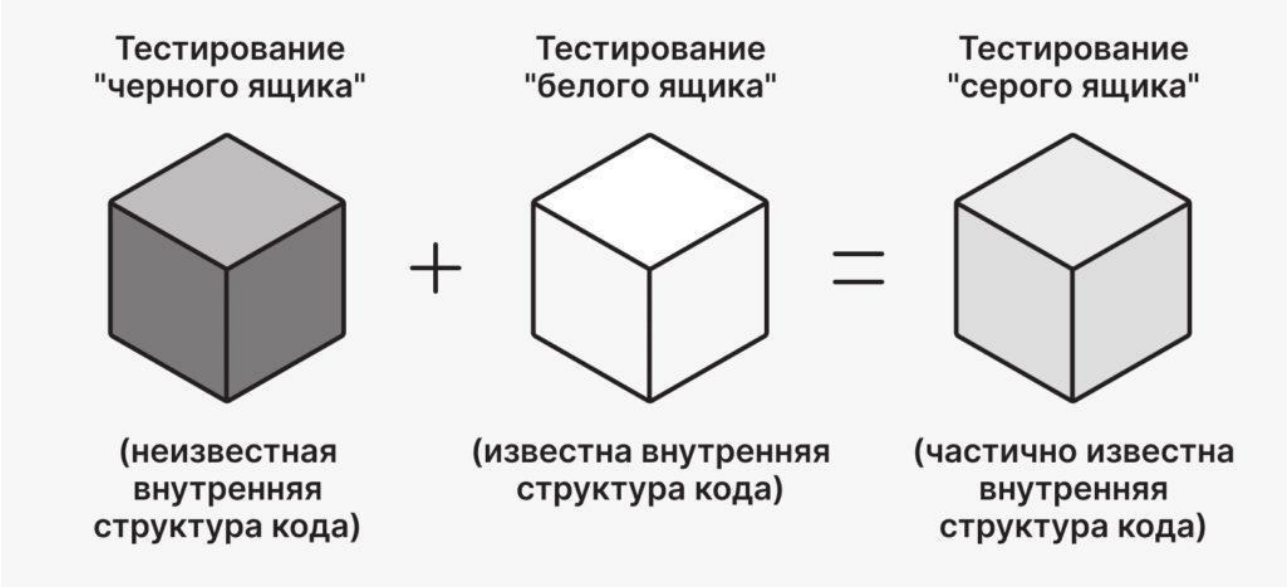

Рис. 3.5 Методы тестирования

Для обеспечения единообразия и структурированности процесса создания тестов часто используется методология, известная как подготовка-действие-проверка (arrange-act-assert). Этот подход позволяет создать четкую и воспроизводимую последовательность шагов.

Подготовка включает в себя подготовку тестового окружения, создание и инициализацию необходимых объектов, а также конфигурацию заглушек.

Действие подразумевает выполнение тестового сценария – вызов конкретного метода с подготовленными тестовыми данными.

Проверка заключается в сравнении полученных результатов с ожидаемыми.

Для обеспечения качества и надежности программного модуля для автоматизированного тестирования веб-форм (ПМ АТВ) был проведен анализ методов и средств тестирования, которые могут быть применены на различных этапах разработки. Тестирование проводилось как для клиентской, так и для серверной частей системы, с учетом их специфики и особенностей взаимодействия.

3.3 Модульное тестирование ПМ АТВ.

Для модульного тестирования программного модуля для автоматизированного тестирования веб-форм (ПМ АТВ) были выбраны следующие библиотеки и инструменты: Jest для тестирования JavaScript-кода клиентской части и pytest для тестирования Python-кода серверной части. Эти инструменты обеспечивает необходимую функциональность для написания и выполнения тестов, а также предоставляют возможности для создания изолированной тестовой среды.

При написании модульных тестов соблюдались следующие принципы:

– изоляция модулей – каждый модуль или компонент тестируется отдельно;

– использование моков – для внешних зависимостей используются моки, что позволяет изолировать тестируемый модуль;

– покрытие позитивных и негативных сценариев – тесты должны проверять как корректные, так и ошибочные сценарии работы модуля;

– покрытие всех критически важных путей выполнения.

Модульное тестирование клиентской части включало проверку основных функций и компонентов, таких как генерация тестовых данных, работа с временными email-адресами, автозаполнением веб-форм и обработка входящих писем через WebSocket.

В таблице 3.1 приведены тест-кейсы модульных тестов для проверки функционала генерации тестовых данных.

Таблица 3.1

Тест-кейсы модульных тестов для функции генерации тестовых данных

№	Тест-кейс	Описание	Ожидаемый результат	Итоговый результат
1	Генерация имени	Проверка корректности генерации имени	Возвращает строку с именем	+
2	Генерация email	Проверка корректности генерации email	Возвращает строку с корректным email-адресом	+
3	Генерация номера телефона	Проверка корректности генерация номера телефона	Возвращает строку с номером телефона	+
4	Генерация адреса	Проверка корректности генерации адреса	Возвращает строку с адресом	+

По данным из таблицы следует, что функция генерации тестовых данных реализована корректно.

В таблице 3.2 приведены тест-кейсы модульных тестов для проверки функционала запроса временного email.

Таблица 3.2

Тест-кейсы модульных тестов для функции генерации email

№	Тест-кейс	Описание	Ожидаемый результат	Итоговый результат
1	Запрос email	Проверка корректности запроса email	Возвращает корректный временный email адрес	+
2	Обработка ответа сервера	Проверка корректности обработки ответа сервера	Обновляет интерфейс с новым email адресом	+

По данным из таблицы следует, что функция генерации временного email реализована корректно.

Модульное тестирование серверной части включало проверку основных классов и функций, таких как генерация временных email-адресов, обработка входящих писем и отправка уведомлений через WebSocket.

В таблице 3.3 приведены тест-кейсы модульных тестов для проверки функции генерации временных email-адресов.

Таблица 3.3

Тест-кейсы модульных тестов для функции генерации email-адресов

№	Тест-кейс	Описание	Ожидаемый результат	Итоговый результат
1	Уникальность email	Проверка уникальности сгенерированного email	Возвращает уникальный адрес email	+

Окончание таблицы 3.3

2	Корректность формата email	Проверка корректности формата email	Возвращает email в корректном формате	+
3	Хранение email в базе данных	Проверка корректности сохранения email в базе данных	Email корректно сохраняется в базе данных	+

По данным из таблицы следует вывод, что функция генерации временных email-адресов реализована корректно.

В таблице 3.4 приведены тест-кейсы модульных тестов для проверки функции обработки входящих писем.

Таблица 3.4

Тест-кейсы модульных тестов для функции обработки писем

№	Тест-кейс	Описание	Ожидаемый результат	Итоговый результат
1	Прасинг письма	Проверка корректности парсинга письма	Возвращает коректно распарсенное письмо	+
2	Сохранение письма в базе данных	Проверка корректности сохранения письма в базе данных	Письмо корректно сохраняется в базе данных	+
3	Уведомление через WebSocket	Проверка корректности отправки уведомлений через WebSocket	Уведомление корректно отправляется через WebSocket	+

По данным из таблицы следует, что функция обработки входящих писем реализована корректно.

Модульные тесты как правило представляют из себя проверку самого базового функционала и реализуются по одному шаблону. На рисунке 3.6 представлены результаты прохождения модульного тестирования серверной части разрабатываемого модуля.

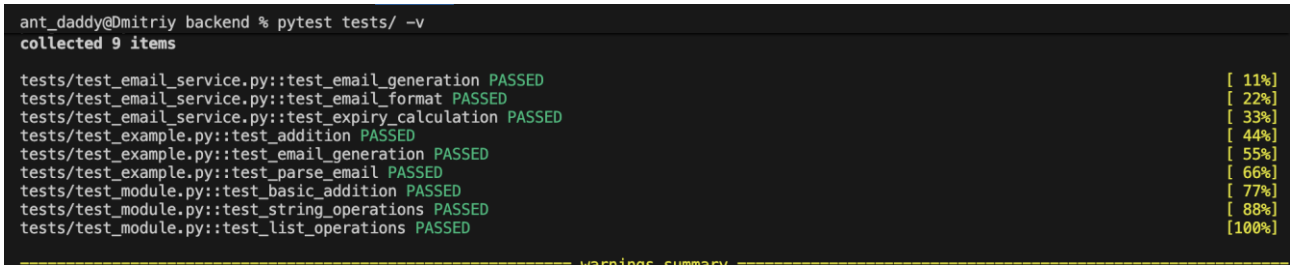

Рис. 3.6 Результаты прохождения модульного тестирования серверной части

Модульное тестирование играет важную роль в обеспечении надежности и стабильности системы, выступая первым уровнем защиты от ошибок. В рамках проведенной работы представлены тест-кейсы для проверки ключевых функций разрабатываемого модуля.

3.4 Интеграционное тестирование ПМ АТВ

Для интеграционного тестирования программного модуля для автоматизированного тестирования веб-форм использовались инструменты, позволяющие эмулировать взаимодействие пользователя с интерфейсом, проверять изменения в DOM и состояние приложения после выполнения различных действий.

Основной шаблон интеграционного теста строится на последовательном выполнении ключевых этапов. Сначала происходит инициализация компонентов и настройка окружения. Затем эмулируются действия пользователя – клики, ввод текста и другие взаимодействия с интерфейсом. После выполнения этих действий анализируются изменения в DOM, чтобы убедиться в корректности отображения и реакции системы. За-

вершающим этапом становится верификация состояния приложения и проверка правильности обработки данных. В таблице 3.5 представлено описание основных тест-кейсов для интеграционных тестов ПМ АТВ.

Таблица 3.5

Тест-кейсы интеграционных тестов ПМ АТВ

№	Тест-кейс	Описание	Шаги воспроизведения	Ожидаемый результат	Итоговый результат
1	Генерация временного email	Проверка генерации временного email-адреса	Открыть Popup; Нажать на кнопку генерации.	Появляется временный email-адрес	+
2	Автозаполнение веб-формы	Проверка автозаполнения веб-формы	Открыть веб-страницу; Нажать на кнопку автозаполнения.	Форма заполняется тестовыми данными	+
3	Получение входящего письма	Проверка обработки входящего письма	Отправить тестовое письмо; Проверить Side Panel.	Письмо отображается в Side Panel	+
4	Обновление Side Panel	Проверка обновления Side Panel при получении письма	Отправить письмо; Проверить обновление.	Side Panel обновляется с новым письмом	+
5	Установка WebSocket-соединения	Проверка установки соединения через WebSocket	Открыть Popup; Проверить соединение.	Соединение успешно установлено	+
6	Обработка ошибок соединения	Проверка обработки ошибок WebSocket соединения	Разорвать соединение; Проверить реакцию системы.	Отображается сообщение об ошибке	+

Окончание таблицы 3.5

7	Взаимодействие с SMTP-сервером	Проверка взаимодействия с SMTP сервером	Отправить письмо; Проверить базу данных.	Письмо сохраняется в базе данных	+
8	Сохранение состояния приложение	Проверка сохранения состояния приложения при обновлении	Сгенерировать данные; Обносить страницу.	Состояние приложения сохраняется	+

По данным из таблицы следует, что взаимодействие между различными частями модуля реализовано корректно.

При тестировании API проверяется, как взаимодействуют различные модули и микросервисы друг с другом, обмениваются данными и реагируют на запросы. В рамках тестирования ПМ АТВ было проведено тестирование API. Тест-кейсы для тестирования API представлены в таблице 3.6.

Таблица 3.6

Тест-кейсы для API эндпоинтов

№	Тест-кейс	Метод	URL/Входные данные	Ожидаемый код	Pass/Fail
1	Создание временного почтового ящика	POST	/api/v1/email -	200	+

Окончание таблицы 3.6

2	Получение информа-ции о ящике	GET	/api/v1/email/{id} Валидный UUID	200	+
3	Получение несуще-ствующего ящика	GET	/api/v1/email/{id} НеВалидный UUID	404	+
4	Получение писем ящика	GET	/api/v1/email/{id}/mes-sages Валидный UUID	200	+
5	Удаление почтового ящика	DELETE	/api/v1/email/{id} Валидный UUID	200	+
6	Получение ящика по адресу	GET	/api/v1/email/by-ad-dres/{email} Существующий email	200	+

Для обоснования корректности работы механизма взаимодействия в реальном времени между клиентской и серверной частями ПМ АТВ были разработаны тест-кейсы, проверяющие функциональность WebSocket-соединений, представленные в таблице 3.7.

Таблица 3.7

Тест-кейсы для WebSocket-соединений

№	Тест-кейс	Действие	Ожидаемый результат	Pass/Fail
1	Установка со-единения	Подключение к ws://localhost:8000/ws/{id}	Получение сообщения «connected»	+

2	Уведомление о новом письме	Отправка письма на ящик	Получение уведомления	+
3	Несуществующий идентификатор	Подключение с невалидным UUID	Закрытие соединения	+
4	Ping/pong механика	Отправка «ping»	Получение «pong» ответа	+

На рисунке 3.7 представлены результаты прохождения интегрального тестирования разрабатываемого модуля.

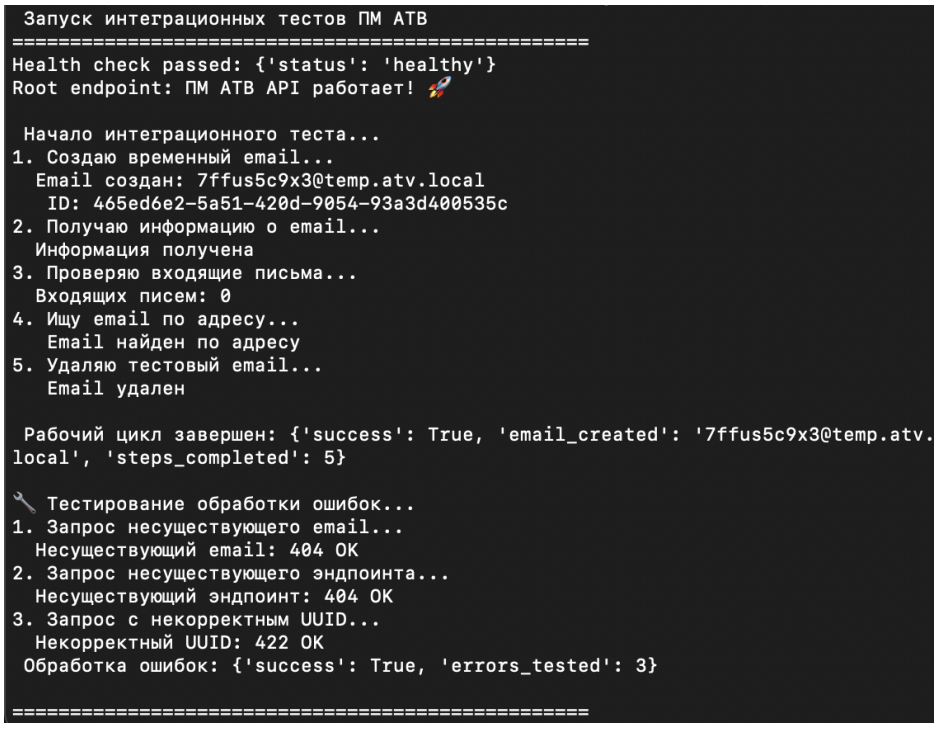

Рис. 3.7 Результаты прохождения модульного тестирования серверной части

Интеграционные тесты проверяют взаимодействие компонентов в условиях, максимально приближенных к реальным. Они охватывают ключевые сценарии: генерацию временных email-адресов, автозаполнение веб-форм и обработку входящих писем, установку WebSocket-соединения.

3.5 Анализ полученных результатов

В условиях современной цифровизации и автоматизации бизнес-процессов особое значение приобретают инструменты, способные оптимизировать процесс тестирования веб-форм. Исследование, проведенное в рамках разработки программного модуля для автоматизированного тестирования веб форм показало, что классические методы тестирования веб-форм требуют значительных временных затрат. Тестировщикам часто приходится вручную заполнять формы, что замедляет процесс тестирования и увеличивает вероятность ошибок.

В ходе тестирования ПМ АТВ были получены данные о повышении показателей эффективности тестирования веб-форм. Наиболее заметные улучшения коснулись временных характеристик: среднее время заполнения стандартной веб-формы сократилось с 5 минут до 1 минуты. Это стало возможным благодаря автоматизации процесса генерации тестовых данных и автозаполнения форм. Качество тестирования также улучшилось за счет уменьшения количества ошибок, связанных с ручным вводом данных.

Снижение количества рутинных операций в процессе тестирования, таких как ручное заполнение форм и проверка входящих писем, позволяет специалистам сосредоточиться на более важных задачах. Сравнительный анализ показателей представлен в таблице 3.8

Таблица 3.8

Сравнительный анализ результатов разработки ПМ АТВ

Критерий оценки	Показатели до внедрения	Показатели после внедрения	Процент оптимизации
Время заполнения стандартной веб-формы	≈ 25 мин	≈ 5 мин	≈ 80%
Количество ошибок заполнения шт./10 форм	≈ 4.5	≈ 0.5	≈ 89%
Время обработки писем	≈ 5 мин	< 1 мин	≈ 80%

Проведенный анализ наглядно демонстрирует, что использование программного модуля для автоматизированного тестирования веб-форм позволяет решить ключевые проблемы тестирования веб-форм. Существенное сокращение временных завтра сочетается с качественным улучшением процесса тестирования за счет автоматизации.

Выводы по разделу

В испытательном разделе были рассмотрены методы и средства для отладки программного модуля для автоматизированного тестирования веб-форм, а также проведено комплексное тестирование, включающее модульное, интеграционное и мануальное тестирование. Это позволило всесторонне проверить корректность работы инструмента на его соответствие установленным требованиям.

В процессе отладки использовались технологии, такие как DevTools, с применением вкладок Elements, Console, Network и Perfomance. Это позволило эффективно выявлять и устранять ошибки на этапе разработки.

Все тестовые сценарии, разработанные для проверки функциональности модуля, были успешно выполнены. Это подтверждает стабильную работу ПМ АТВ в различных условиях эксплуатации. Были проверены такие ключевые сценарии, как генерация временных email-адресов, автозаполнение веб-форм, обработка входящих писем и взаимодействие с серверной частью через WebSocket.

Использование ПМ АТВ позволило оптимизировать рабочие процессы, снизить нагрузку на тестировщиков, а также повысить скорость и качество тестирования веб-форм.

ЗАКЛЮЧЕНИЕ

Результатом выпускной квалификационной работы является рабочая версия программного модуля для автоматизированного тестирования веб-форм. Основной целью проекта является сокращение временных затрат на тестирование веб-форм и повышение работы специалистов по тестированию.

В процессе выполнения работы было проведено глубокое исследование предметной области, а также выполнен сравнительный анализ существующих решений. Это позволило сформировать требования к функциональным возможностям модуля. На основе анализа были разработаны концептуальная модель и алгоритм работы модуля, которые включают механизмы генерации тестовых данных, управления временными email-адресами, автозаполнения веб-форм и обработки входящих писем.

Для реализации ПМ АТВ были использованы современные технологии разработки. Клиентская часть была разработана с использованием JavaScript в виде браузерного расширения. Серверная часть была реализована на Python с использованием фреймворка FastAPI. Для обеспечения двусторонней связи между клиентской и серверной частями в реальном времени использовался протокол WebSocket. Для реализации функциональности SMTP сервера была использована библиотека SMTPlib.

Время заполнения веб-форм сократилось на 80%, что ускорило процесс тестирования. Количество ошибок заполнения значительно уменьшилось, что повысило надежность и точность тестирования. Время обработки входящих писем сократилось на 80%. Нагрузка на тестировщиков снизилась за счет автоматизации рутинных операций.

Таким образом, цель разработки была достигнута, и был создан эффективный инструмент, который ускоряет процесс тестирования веб-форм и повышает его качество.

В перспективе планируется дальнейшее развитие функционала ПМ АТВ. Будет добавлена возможность гибких настроек внутри расширения и возможность совместной работы для командного использования модуля. Эти улучшения позволят сделать ПМ АТВ более универсальным и эффективным решением для тестирования веб-форм.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. ГОСТ Р 7.0.5-2008. Система стандартов по информации, библиотечному и издательскому делу. Библиографическая ссылка. Общие требования и правила составления. Введ. 01.01.2009. М. : Стандартинформ, 2020. 24 с
2. Гайдук И. О., Касимов Р. А., Кононова А. И., Федоров А. Р., Федотова Е. Л. Методические указания по подготовке выпускной квалификационной работы по направлению подготовки бакалавров 09.03.04 «Программная инженерия» / под ред. Л. Г. Гагариной. М. : МИЭТ, 2024. 36 с.
3. Федеральный государственный образовательный стандарт высшего образования для бакалавриата по направлению 09.03.04 «Программная инженерия». М., 2017.
4. ГОСТ 19.101-77. Единая система программной документации. Виды программ и программных документов. Введ. 01.01.1980. М. : Стандартинформ, 2010. 3 с.
5. ГОСТ 19.401-78. Единая система программной документации. Текст программы. Требования к содержанию и оформлению. Введ. 01.01.1980. М. : Стандартинформ, 2010. 2 с.
6. ISTQB. Официальный веб-сайт [Электронный ресурс]. 2025. URL: https://istqb.org/
7. ГОСТ Р ИСО/МЭК 12207-99. Процессы жизненного цикла программных средств. Введ. 01.07.2000. М. : Стандартинформ, 2003. 2 с.
8. ГОСТ Р ИСО/МЭК 25010-2015. Требования и оценка качества систем и программного обеспечения (SQuaRE). Модели качества систем и программных продуктов. Введ. 01.06.2016. М. : Стандартинформ, 20018. 2 с.
9. Capgemini. Официальный веб-сайт [Электронный ресурс]. 2023. URL: https://www.capgemini.com/insights/research-library/world-quality-report-2023-24/
10. Gratner. Официальный веб-сайт [Электронный ресурс] - URL: https://www.gartner.com/ru/insights
11. Документация GuerrillaMail. [Электронный ресурс]. - URL: https://www.guerrillamail.com/ru/
12. Документация TempMail. [Электронный ресурс]. - URL: https://temp-mail.org

13. Документация Selenium. [Электронный ресурс]. - URL: https://www.selenium.dev/
14. Документация Faker. [Электронный ресурс]. - URL: https://faker.readthedocs.io/
15. Документация Mailtrap. [Электронный ресурс]. - URL: https://mailtrap.io/
16. Документация Java. [Электронный ресурс]. - URL: https://www.java.com/ru/
17. Документация C#. [Электронный ресурс]. - URL: https://dotnet.microsoft.com/ru-ru/languages/csharp
18. Документация Golang. [Электронный ресурс]. - URL: https://go.dev/
19. Документация Node.js. [Электронный ресурс]. - URL: https://nodejs.org/en
20. Документация Python. [Электронный ресурс]. - URL: https://www.python.org/
21. Документация TypeScript. [Электронный ресурс]. - URL: https://www.typescriptlang.org/
22. Документация Dart. [Электронный ресурс]. - URL: https://dart.dev/
23. Документация WebAssembly. [Электронный ресурс]. - URL: https://webassembly.org/
24. Документация JavaScript. [Электронный ресурс]. - URL: https://learn.javascript.ru/
25. Документация PyCharm. [Электронный ресурс]. - URL: https://www.jetbrains.com/pycharm/
26. Документация WebStorm. [Электронный ресурс]. - URL: https://www.jetbrains.com/ru-ru/webstorm/
27. Документация IntelliJ IDEA. [Электронный ресурс]. - URL: https://www.jetbrains.com/idea/
28. Документация Visual Studio Code. [Электронный ресурс]. - URL: https://code.visualstudio.com/
29. Документация по фронтенд-тестированию. [Электронный ресурс]. - URL: https://habr.com/ru/articles/838986/
30. Документация по тестированию. [Электронный ресурс]. - URL: https://testengineer.ru/

31. ГОСТ 7.89-2005. Система стандартов по информации, библиотечному и издательскому делу. Оригиналы текстовые авторские и издательские. Общие требования. - Введ. 01.07.2006. - М.: Стандартинформ, 2006. - 18 с.

32. ГОСТ 19.701-90 (ИСО 5807-85). Единая система программной документации. Схемы алгоритмов, программ, данных и систем. Обозначения условные и правила выполнения. - Введ. 01.01.1992. - М.: Стандартинформ, 2010. - 24 с.

33. ГОСТ 15971-90. Системы обработки информации. Термины и определения. - Введ. 01.01.1992. - М.: Изд-во стандартов, 1991. - 14 с.

34. ГОСТ 28806-90. Качество программных средств. Термины и определения. - Введ. 01.01.1992. - М.: Стандартинформ, 2005. - 8 с.

35. ГОСТ Р 53622-2009. Информационные технологии. Информационно-вычислительные системы. Стадии и этапы жизненного цикла, виды и комплектность документов. - Введ. 01.01.2011. - М.: Стандартинформ, 2019. - 12 с.