

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»

Профиль подготовки: «Вычислительные машины, комплексы, системы и сети»

БАКАЛАВРСКАЯ РАБОТА

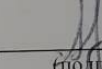
на тему

МОДУЛЬ ВЕБ-ПРИЛОЖЕНИЯ ДЛЯ
АВТОМАТИЗИРОВАННОГО СБОРА И
СИСТЕМАТИЗАЦИИ ДАННЫХ О
НЕБЛАГОНАДЕЖНЫХ КЛИЕНТАХ БАНКА

Студент

 04.06.2025 Карев Д. Ф.
(подпись, дата)

Руководитель

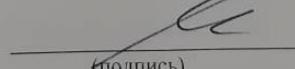
 04.06.2025 Акифьев И. В.
(подпись, дата)

Нормоконтролёр

 04.06.2025 Бычков А. С.
(подпись, дата)

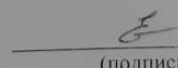
Работа допущена к защите (протокол зас. кафедры от 28.05.2025 г. № 16)

Заведующий кафедрой

 М.А. Митрохин
(подпись)

Работа защищена с отметкой хорошо (протокол заседания ГЭК от 11.06.25 № 12)

Секретарь ГЭК

 Е.И. Гурин
(подпись)

Пенза 2025

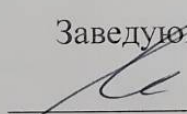
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»

Заведующий кафедрой

 М.А. Митрохин

«06» мая 2025 г.

**ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ**

1. Студент Карев Данила Фёдорович гр. 21BBV1 факультета вычислительной техники.
2. Тема работы «Модуль веб-приложения для автоматизированного сбора и систематизации данных о неблагонадежных клиентах банка»

Тема утверждена приказом ПГУ № 601/0 от "06" мая 2025 г.

3. Руководитель бакалаврской работы к.э.н., доцент, Акифьев И. В.

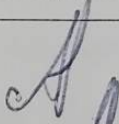
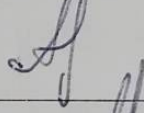
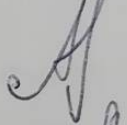
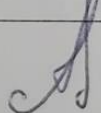
4. Задание на работу (назначение разработки, исходные данные и т.п.)

Разработать “Модуль веб-приложения для автоматизированного сбора и систематизации данных о неблагонадежных клиентах банка”, выполняющее сбор данных с АСОИ ФинЦЕРТ, анализ полученных табличных данных и их обработку. Приложение должно быть спроектировано с использованием слоистой архитектуры. Программа должна обладать графическим интерфейсом. Система должна включать в себя функционал сбора и сохранения данных в базу данных. Также программа должна загружать обработанные данные на удаленный SFTP-сервер. Вывод всей полученной информации необходимо выводить с помощью веб-приложения.

5. Перечень подлежащих разработке вопросов

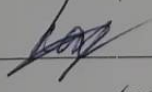
Требуется произвести анализ предметных областей: изучить существующие аналоги, изучить актуальные способы анализа и хранения табличных данных, изучить способы проектирования и разработки собственного ПО с применением REST API методологии, изучить методы работы с сертификатами и ключами, разобраться, как взаимодействовать с SFTP-сервером. Также необходимо изучить функционал фреймворков Django и .NET Framework.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Изучение предметной области	52	06.05.2025-09.05.2025	
2.	Анализ требований	13	10.05.2025	
3.	Выбор технологий, применяемых при проектировании и разработке	52	11.05.2025-14.05.2025	
4.	Описание серверной части приложения	39	15.05.2025-17.05.2025	
5.	Разработка модуля веб-приложения для автоматизированного сбора и систематизации данных	104	18.05.2025-25.05.2025	
6.	Оформление пояснительной записки	64	26.05.2025-30.05.2025	

Дата выдачи задания " 06 " мая 20 25 г.

Руководитель бакалаврской работы  06.05.2025 Акифьев И. В.
(подпись, дата)

Задание к исполнению принял студент  06.05.2025 Карев Д. Ф.
(подпись, дата)

Работу к защите допустить
Декан факультета  09.06.2025 Фионова Л. Р.
(подпись, дата)

Реферат

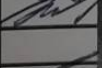
Бакалаврская работа содержит 125 листов, 31 рисунков, 19 таблиц, 7 источников и 3 приложения.

АВТОМАТИЗИРОВАННАЯ СИСТЕМА, ОБРАБОТКА СОБЫТИЙ, ЛОГИРОВАНИЕ, БАЗА ДАННЫХ, ПАРАЛЛЕЛЬНОЕ ПРОГРАММИРОВАНИЕ, C#, WINDOWS FORMS, POSTGRESQL, PYTHON, .NET FRAMEWORK, DJANGO.

Цель работы — разработка модуля веб-приложения для автоматизированного сбора и систематизации данных о неблагонадежных клиентах банка с целью упрощения процесса загрузки данных с внешнего ресурса ФинЦЕРТ.

Для достижения поставленной цели было разработано приложение, отвечающим следующим поставленным задачам: настройка файлов конфигурации, с использованием графического интерфейса, аутентификация на внешнем ресурсе ФинЦЕРТ по средством REST API, обработка полученных данных, загрузка полученных файлов в БД, загрузка полученных файлов на SFTP-сервер, отображение таблиц с полученными данными на развернутом веб-приложении.

В результате проведенной работы разработан модуль веб-приложения для автоматизированного сбора и систематизации данных о неблагонадежных клиентах банка с графическим интерфейсом Windows Forms для работы с файлами конфигурации, слоистой архитектурой на языке C#, веб-приложение на языке Python. Для хранения данных применена СУБД PostgreSQL в связке с графическим клиентом pgAdmin4.

					ПГУ 1.090301.09.001 ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата				
Разраб.		Карев Д. Ф.		04.06.25	Модуль веб-приложения для автоматизированного сбора и систематизации данных о неблагонадежных клиентах банка. Пояснительная записка	Лит.	Лист	Листов
Провер.		Акифьев И.В.		04.06.25				127
Ил. Контр		Бычков А.С.		04.06.25		ФВТ, каф. ВТ		
Утверд.		Митрохин М.А.		04.06.25		гр. 21BBB1		

Содержание

Введение.....	4
1 Модуль веб-приложения для автоматизированного сбора и систематизации данных о неблагонадежных клиентах банка.....	5
1.1 Анализ предметной области приложения	5
1.1.1 Актуальность разработки.....	5
1.1.2 Анализ существующих аналогов	6
1.1.3 Постановка задачи	9
1.2 Выбор технологий и средств разработки приложения	10
1.2.1 Выбор языка программирования.....	10
1.2.2 Выбор СУБД.....	11
1.2.3 Графический клиент pgAdmin 4.....	13
1.2.4 Выбор среды разработки.....	13
1.2.5 WinForms API для создания графического интерфейса	15
1.2.6 Выбор фреймворка для создания веб-приложения	15
1.3 Проектирование информационной системы	17
1.3.1 Архитектура приложения.....	17
1.3.2 Проектирование базы данных	20
1.4 Разработка приложения	35
1.4.1 Реализация БД	35
1.4.2 Описание программы	43
1.4.3 Описание веб-приложения.....	62
1.4.4 Руководство пользователя	67
2 Охрана труда.....	79
Заключение	82
Список использованных источников	83
Приложение А – UML-Диаграммы	84
Приложение Б – Листинг программы	89
Приложение В - Презентация	122

Перечень принятых сокращений и обозначений

АС - автоматизированная система

БД — база данных

ОС — операционная система

СУБД — система управления базами данных

Многопоточность — способность программы выполнять несколько потоков (threads) одновременно

API — описание способов, которыми одна компьютерная программа может взаимодействовать с другой программой

FK — внешний ключ, понятие теории реляционных баз данных, относящееся к ограничениям целостности базы данных

GUI — графический интерфейс пользователя

ID — уникальный номер объекта

IDE — интегрированная среда разработки

JSON — текстовый формат обмена данными

PK — первичный ключ, понятие теории реляционных баз данных, относящееся к ограничениям целостности базы данных

SQL — язык структурированных запросов

UML — унифицированный язык моделирования

					ПГУ 1.090301.09.001 ПЗ	Лист
						3
Изм.	Лист	№ докум.	Подпись	Дата		

Введение

Сегодня банки и финансовые компании сталкиваются с серьезной проблемой - им нужно постоянно проверять клиентов на возможную причастность к отмыванию денег или финансированию терроризма. Эти проверки требуют огромного количества времени и ресурсов, если делать их вручную. Представьте себе: каждый день в банк приходят новые клиенты, совершаются тысячи операций. Сотрудникам нужно проверять всех и вся по специальным спискам, которые к тому же регулярно обновляются. Если что-то пропустить - банк может получить крупный штраф или даже лишиться лицензии.

Центральный Банк России создал специальный сервис - API ФинЦерт. Это как "окно" в базу данных всех подозрительных лиц и организаций. Но чтобы этим удобно пользоваться, нужно специальное программное обеспечение.

Такое программное обеспечение должно: автоматически получать свежие данные из ФинЦерт, аккуратно хранить их в базе данных. Данное ПО, сможет помочь банкам быстрее проверять клиентов, снизится вероятность ошибок, всегда будут под рукой актуальные данные.

Результатом станет готовая к использованию система, которую банки смогут внедрить у себя для автоматизации проверок клиентов. Это сделает финансовый сектор более безопасным и прозрачным, а работу банков - более эффективной.

					ПГУ 1.090301.09.001 ПЗ	Лист
						4
Изм.	Лист	№ докум.	Подпись	Дата		

1 Модуль веб-приложения для автоматизированного сбора и систематизации данных о неблагонадежных клиентах банка

1.1 Анализ предметной области приложения

1.1.1 Актуальность разработки

В наше время все финансовые организации требуют постоянного мониторинга данных о выявленных подозрительных операциях, такие как: транзакции, санкционные списки, изменения статусов юрлиц. Производить постоянный отчет данных для Центрального Банка, Росфинмониторинга и других органов.

Сейчас на себя эту роль взяла ФинЦерт (национальная система финансового мониторинга). ФинЦерт — это российская государственная система, занимающаяся мониторингом и противодействием киберугрозам в финансовой сфере. Она работает под эгидой Банка России и играет ключевую роль в обеспечении безопасности платежных систем, банков и других финансовых организаций.

Основные задачи ФинЦерт:

- обнаружение кибератак на банки, платежные системы и финтех-компании;
- анализ фишинговых атак, DDoS-нападений, мошеннических схем (например, поддельные мобильные приложения банков);
- распространение предупреждений о новых угрозах через фиды (API, рассылки, порталы);
- координация реагирования на инциденты (например, блокировка вредоносных IP-адресов).

ФинЦерт предоставляет данные со списками вредоносных URL, IP адресов и доменов; тактики мошенников, рекомендации по защите; санкционные списки. Данные приходят в различных форматах: JSON/XML при

					ПГУ 1.090301.09.001 ПЗ	Лист
						5
Изм.	Лист	№ докум.	Подпись	Дата		

работе с API, CSV-отчеты для ручной загрузки и экстренные оповещения через телеграмм боты.

С ФинЦерт обязаны работать абсолютно все банки, которые находятся на финансовом рынке, различные страховые компании и некоторые платежные системы. За игнорирование все компании понесут огромные штрафы от Банка России и будут подвержены риску хищения денежных средств клиентов.

Немного подумав можно сделать вывод, что ФинЦерт — это «щит» российской банковской системы.

1.1.2 Анализ существующих аналогов

Существует несколько аналогов ФинЦерт, которые помогают бороться с киберугрозами, мошенничеством и отмыванием денег.

Внешний вид сервиса «Cyber Polygon» представлен на рисунке 1.

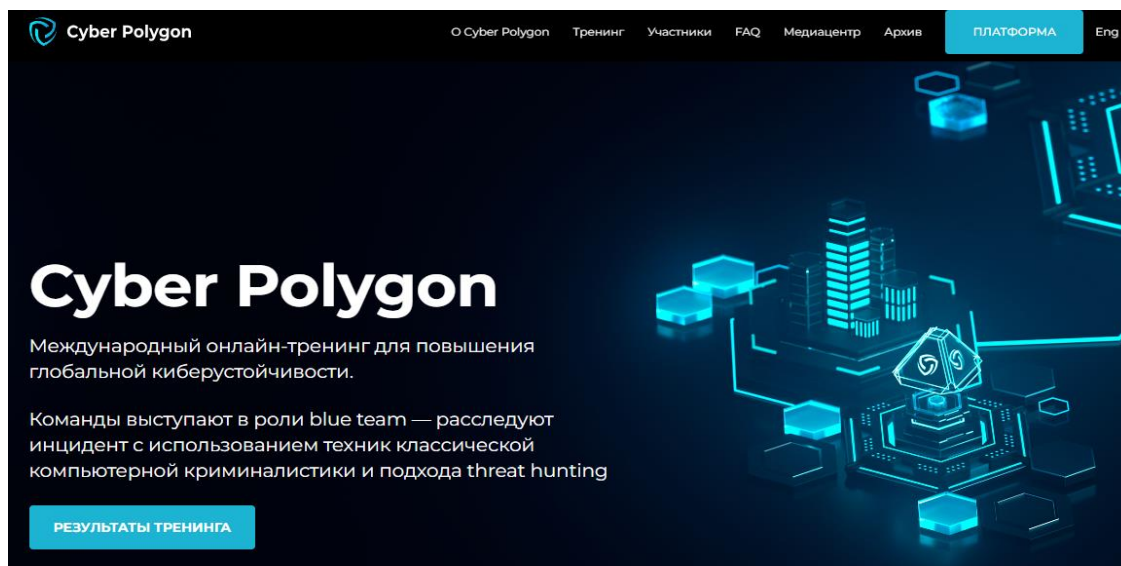


Рисунок 1 - Внешний вид «Cyber Polygon».

Разработчик: Центральный банк Российской Федерации.

Ссылка: <https://cyberpolygon.com/ru/>.

Платформа для тестирования киберустойчивости финансовых организаций.

Плюсы:

- тестирование уязвимостей финансовых организаций;
- абсолютно бесплатное пользование для всех участников

финансового рынка.

Из минусов можно выделить главный – это то, что он не может полностью заменить ФинЦЕРТ, так как ориентирован на тестирование, а не на оперативный мониторинг.

Внешний вид сервиса «Angara Security» представлен на рисунке 2.

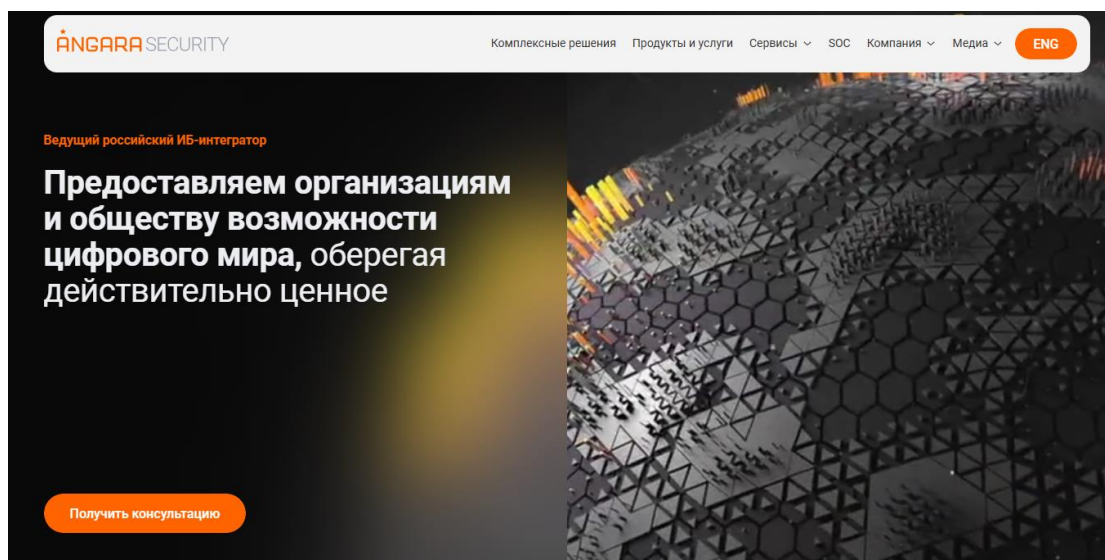


Рисунок 2 - Внешний вид «Angara»

Разработчик: Positive Technologies.

Ссылка: <https://www.angarasecurity.ru/>.

Коммерческая платформа для защиты от кибератак.

Плюсы:

- имеет готовую аналитику и автоматическое реагирование;
- поддерживает международные стандарты.

Минусы:

- для использования необходимо иметь платную подписку или осуществлять разовые платежи;
- не может полностью заменить ФинЦЕРТ.

Внешний вид сервиса «Антифрод-система FraudWall» представлен на рисунке 3.

					ПГУ 1.090301.09.001 ПЗ	Лист
						7
Изм.	Лист	№ докум.	Подпись	Дата		

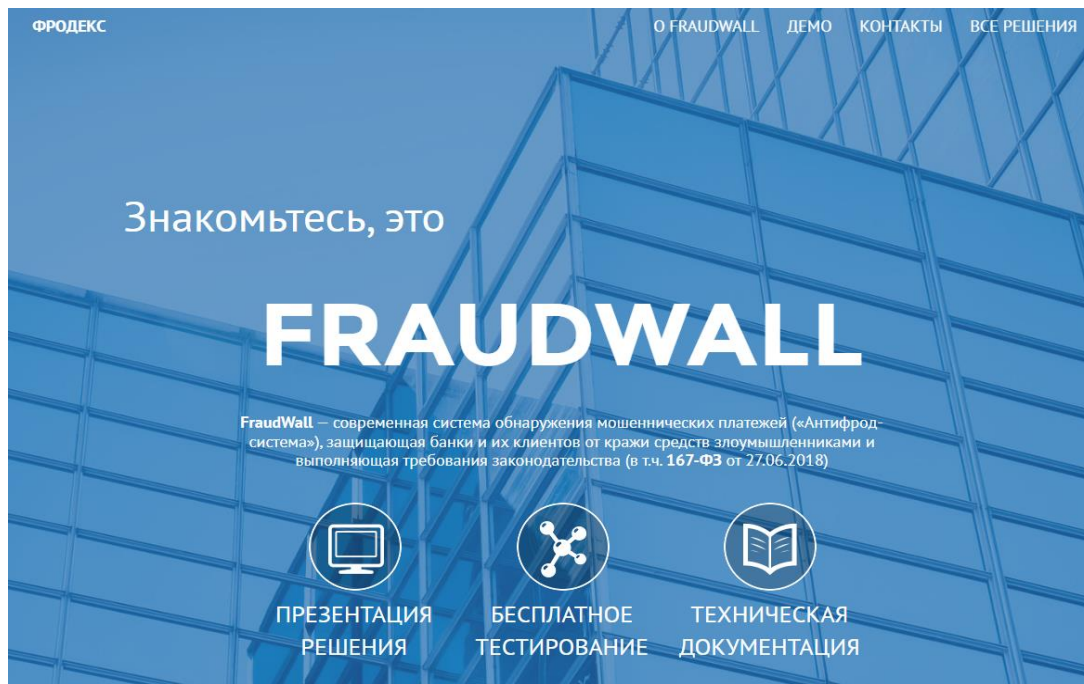


Рисунок 3 - Внешний вид «FraudWall»

Разработчик: «Фродекс».

Ссылка: <https://www.fraudwall.ru/>.

Современная система обнаружения мошеннических платежей.

Плюсы:

- проверяет клиентов на санкции;
- имеет готовые интеграции (1С, банковские системы);
- имеет поддержку международных стандартов.

Минусы:

- платное взаимодействие с ресурсом;
- не предоставляет данные о кибератаках;
- изменение в API, могут испортить всю интеграцию.

1.1.3 Постановка задачи

Требуется разработать программное обеспечение, которое позволяет автоматизировать процесс получения, обработку и хранения данных о неблагонадежных клиентах банков с сервиса ФинЦЕРТ.

Приложение должно отвечать следующим требованиям:

- возможность изменять файлы конфигурации с помощью графического интерфейса пользователя;
- возможность устанавливать защищенное соединение для передачи данных;
- возможность обрабатывать полученные данные для дальнейшего хранения в базе данных;
- возможность настраивать работу программы в определенное время с помощью планировщика задач;
- создание лог-файлов (файлы хранящие информацию о событиях, происходящих во время работы программы);
- дата и время должны выводиться на веб-сервер для визуализации;
- клиенты должны иметь возможность просмотра даты последнего скачивания данных.

Разрабатываемая автоматизированная система должна соответствовать следующим техническим требованиям:

- операционная система: Windows 7 и более поздние версии;
- оперативная память: не менее 512 МБ ОЗУ;
- незанятое дисковое пространство: не менее 512 МБ ПЗУ;
- установленная программа КриптоПро CSP для установки соединения TLS;
- сертификат TLS клиента и цепочка доверия;
- необходимо иметь логин и пароль участника ЦБ России.

					ПГУ 1.090301.09.001 ПЗ	Лист
						9
Изм.	Лист	№ докум.	Подпись	Дата		

1.2 Выбор технологий и средств разработки приложения

В данном разделе будут описаны технологии и средства разработки, использованные при разработке программы.

Ниже приведен список выбранных технологий и средств разработки:

- C#;
- .NET Framework;
- Visual Studio 2022;
- PostgreSQL;
- pgAdmin4;
- Python;
- Django framework.

1.2.1 Выбор языка программирования

Язык программирования C++.

C++ - это язык программирования высокого уровня, разработанный в 1985 году и ставший основой для многих современных технологий, таких как игровые движки, веб-браузеры, операционные системы и т. д. Программы на C++ имеют богатую поддержку библиотек и разнообразие типов данных. Обладает возможностью объектно-ориентированных языков. Он также поддерживает многопоточность и множество парадигм программирования. Однако C++ имеет сложность с управлением памятью, сложность с управлением параллельностью.

Язык программирования Java.

Java – это объектно-ориентированный язык программирования, основанный на классах, чтобы иметь как можно меньше зависимостей от реализации. Программы написанные на Java могут запускаться на любом устройстве благодаря встроенной виртуальной машине (JVM). Java обладает, автоматическим управлением памятью, механизмом сборки мусора. Однако Java имеет проблемы с производительностью, она использует больше памяти и работает медленнее других языком программирования.

					ПГУ 1.090301.09.001 ПЗ	Лист
						10
Изм.	Лист	№ докум.	Подпись	Дата		

Язык программирования C#.

C# - это объектно-ориентированный язык программирования, созданный компанией Microsoft, который работает под управлением на платформе .NET Framework. C# имеет автоматическое управление памятью, что позволяет создавать высокоэффективные программы. Также язык имеет статическую типизацию, что обеспечивает большую надежность кода и позволяет выявлять ошибки на этапе компиляции. Имеет обширную библиотеку для разработки под любые задачи.

После сравнения возможностей одних из популярных языков для реализации приложения был выбран C#, так как он имеет лучшую интеграцию с Windows. Также язык имеет мощный инструмент LINQ (Language Integrated Query) для работы с коллекциями и данными. Данный инструмент отличной подойдет для обработки полученных данных.

Язык программирования Python.

Python – это объектно-ориентированный язык программирования высокого уровня, используемый для создания ряда веб-сайтов и программного обеспечения в различных отраслях, включая анализ данных, машинное обучение, веб-разработку, искусственный интеллект и т. д. Python предлагает большую стандартную библиотеку, которая предлагает богатый набор пакетов и модулей и оснащена необходимыми функциями. Python поддерживает различные парадигмы программирования, и объектно-ориентированное программирование является одной из них.

1.2.2 Выбор СУБД

СУБД PostgreSQL.

В качестве СУБД была выбрана PostgreSQL – это база данных с открытым исходным кодом, которая имеет прочную репутацию благодаря своей надежности, гибкости и поддержке открытых технических стандартов.

Среди преимуществ PostgreSQL можно выделить:

- кроссплатформенность;
- возможность работать с большими размерами данных;

- наследование;
- наличие достаточно подробной документации;
- расширяемость и богатый набор типов данных.

Главным фактором при выборе данной СУБД была его кроссплатформенность, что позволяет использовать ее на любой необходимой платформе. Также PostgreSQL имеет различные графические библиотеки и приложения для более удобной работы с ним.

СУБД MySQL.

MySQL – это одна из самых популярных реляционных СУБД с открытым исходным кодом.

Преимущества MySQL:

- бесплатна для использования;
- обладает хорошей производительностью и способна эффективно обрабатывать большие объемы данных.

Однако стоит помнить, что MySQL не всегда поддерживает хранимые триггеры так эффективно, как другие СУБД. Также могут возникнуть сложности с масштабированием БД на большие объемы данных.

СУБД SQLite.

SQLite – это встраиваемая (файловая) реляционная система управления базами данных (СУБД), которая хранит всю базу данных в одном файле на диске и не требует отдельного серверного процесса.

Преимущества SQLite:

- не требует установки сервера;
- вся база данных хранится в одном файле, что упрощает перенос и резервное копирование;
- не нужно настраивать пользователей, права доступа или сетевые подключения;
- отлично подходит для небольших приложений или проектов с невысокой нагрузкой.

Стоит помнить, что SQLite не имеет клиент-серверной архитектуры, что не подходит для высоконагруженных веб-приложений. Имеет ограниченную конкурентность — только один писатель в момент времени.

1.2.3 Графический клиент pgAdmin 4

pgAdmin4 – это графический интерфейс (GUI) для управления PostgreSQL, бесплатный и открытый инструмент, разработанный для администрирования, разработки и мониторинга баз данных PostgreSQL.

К основным достоинствам pgAdmin4 можно отнести:

- позволяет создавать, редактировать и удалять БД, таблицы, индексы, представлений, функции и триггеры;
- имеет визуальный конструктор запросов с подсветкой синтаксиса SQL;
- поддерживает сложные запросы, включая оконные функции и JSON-операции;
- имеет возможность сохранять часто используемые запросы;
- позволяет вести просмотр статистики производительности (активные сессии, нагрузки на сервер);
- возможность создавать роли и настраивать привилегии;
- поддерживает SSL-шифрование для безопасного подключения.

1.2.4 Выбор среды разработки

Среда разработки Visual Studio 2022.

Visual Studio 2022 – это интегрированная среда разработки (IDE) от Microsoft, предназначенная для создания приложений на C#, C++, Visual Basic и других языках. Это самая современная и мощная версия Visual Studio \, оптимизированная для разработки под Windows, macOS, Linux, Android, IOS и веб- платформы.

Visual Studio 2022 имеет ряд достоинств, упрощающих разработку программного обеспечения:

- поддерживает .NET 6/7/8, включая .NET Core, .NET Framework, также имеет готовые шаблоны проектов;

- имеет умный редактор кода IntelliSense (автодополнение кода с учетом контекста), IntelliCode (AI-подсказки на основе популярных репозиториях GitHub), CodeLens (отображение истории изменений, тестов и ссылок на код), рефакторинг (быстрое преименование, извлечение методов, оптимизация кода);

- имеет улучшенный отладчик с возможностью просмотра сложных объектов (деревья, JSON, XML) и отладка в многопоточных и асинхронных сценариях. Позволяет диагностировать производительность с помощью профилировщика CPU и памяти, и анализировать использования сети и диска.

Данная среда разработки была выбрана потому, что обладает интуитивно понятным интерфейсом, легка в установке и настройке, имеет встроенный графический редактор, упрощающий создание интерфейса приложения.

Среда разработки Visual Studio Code.

Visual Studio Code – это мощный и гибкий редактор кода, который отлично подходит для разработки на многих языках программирования, особенно если нам нужна легковесная альтернатива других IDE. Благодаря расширениям и встроенным инструментам, VS Code обеспечивает почти IDE-уровень поддержки C#, включая IntelliSense, отладку, рефакторинг и работу с .NET.

Преимущества Visual Studio Code:

- потребляет меньше ресурсов, чем многие другие IDE;
- кроссплатформенность, а именно работает на Windows, macOS и Linux;
- с помощью библиотеки расширений можно настроить под любой язык программирования.

Главными недостатками Visual Studio Code является то, что меньше встроенных инструментов для сложной отладки. Не имеет встроенного дизайнера WinForms/WPF (как в Visual Studio).

Исходя из выше перечисленных недостатков, был сделан выбор в пользу среды разработки Visual Studio 2022.

1.2.5 WinForms API для создания графического интерфейса

Создание приложений с графическим интерфейсом пользователя (GUI) в C# эффективно реализуется с помощью технологии Windows Forms – мощного инструмента для разработки в системе .NET. В отличие от Win32 API или MFC, которые требуют ручного управления ресурсами и сложной работы с сообщениями ОС, Windows Forms предоставляет единую объектно-ориентированную модель, автоматизирующую многие аспекты разработки. Это снижает количество ошибок и ускоряет процесс создания интерфейсов. Приложения на Windows Forms выполняются в среде Common Language Runtime, что обеспечивает автоматическое управление памятью и безопасность типов на этапе компиляции.

.NET Framework Class Library включает готовые элементы управления (кнопки, списки, таблицы и др.), которые можно настраивать без низкоуровневого кодирования:

- PropertyGrid для редактирования объектов в формате JSON;
- DataGridView для работы с табличными данными.

В данном проекте, как раз отлично будет подходить PropertyGrid для создания окна для изменения конфигурации приложения.

1.2.6 Выбор фреймворка для создания веб-приложения

Очень важно реализовать удобный вывод информации из базы данных для сотрудников банка, для удобного анализа данных и мониторинга информации. Вывод было решено выводить на удаленный Web-сервер, чтобы каждый сотрудник, мог ознакомиться с информацией на ней не скачивая дополнительное ПО.

Фреймворк Flask.

Flask – это облегченный фреймворк для веб-приложения WSGI. Он разработан для быстрого и простого запуска с возможностью масштабирования до сложных приложений.

Достоинства Flask:

- предоставляет разработчикам независимость и контроль над архитектурой приложения, библиотеками и расширениями;
- имеет возможность обрабатывать запросы (HTTP, RESTful);
- легко интегрируется с любым стилем работы, что особенно подходит опытным разработчикам.

Главный недостаток Flask является, это большая затрата времени на настройку и разработку своего приложения, так как нет встроенной аутентификации, Flask не имеет админ панели. Самый важный недостаток - это отсутствие встроенной системы миграции БД.

Фреймворк Django.

Django – это full-stack фреймворк на языке программирования Python. Он был создан для более простого создания сложных веб-сайтов и приложений с поддержкой баз данных. Django включает в себя большое количество встроенных инструментов и функций, что позволяет сосредоточиться на разработке, а не на настройке инфраструктуры.

Преимущества Django:

- подходит для создания большого количества разнообразных проект, включая системы управления контентом (CMS), социальные сети и научные платформы;
- предоставляет разработчику большое количество встроенных функций, такие как кэширование, административную панель, карты сайта, аутентификация и многое другое;
- предлагает надежные системы аутентификации пользователей и защищает от распространенных угроз, таких как SQL-инъекций.

Учитывая вышесказанные преимущества, выбор был сделан в пользу Django, так как он отлично подходит для работы с базами данных, имеет высокую безопасность, что особенно важно для структуры банка.

					ПГУ 1.090301.09.001 ПЗ	Лист
						16
Изм.	Лист	№ докум.	Подпись	Дата		

1.3 Проектирование информационной системы

1.3.1 Архитектура приложения

В данном проекте была организована слоистая архитектура приложений (Layered Architecture). Слоистая архитектура — это один из самых распространенных подходов к проектированию программного обеспечения, при котором система разбивается на логические слои, каждый из которых выполняет строго определенную функцию.

Первый слой — это слой представления. Он отвечает за представление данных и взаимодействие с пользователем. На этом уровне пользователи смогут с помощью пользовательского интерфейса изменять конфигурацию приложения, а также просматривать веб-сайт, в котором будут храниться обработанные данные.

Второй слой — это слой бизнес-логики. Этот слой связующее звено между слоем представления и слоем доступа к данным. Главная задача данного уровня — обработка бизнес-правил, запросов и операций, связанных с данными. На данном уровне будет происходить взаимодействия нашего приложения с API ФинЦЕРТ, а именно: создавать и отправлять запросы на аутентификацию, запрашивать данные для скачивания. Также здесь будет происходить обработка, полученных данных перед их дальнейшей отправкой.

Третий слой — это слой интеграции. Главная задача данного слоя — это взаимодействие с внешними системами и базами данных. На данном уровне будет осуществляться взаимодействие с базой данных, также обработанные данные будут загружаться на межсетевой экран.

На рисунке 4 представлена диаграмма развертывания.

					ПГУ 1.090301.09.001 ПЗ	Лист
						17
Изм.	Лист	№ докум.	Подпись	Дата		

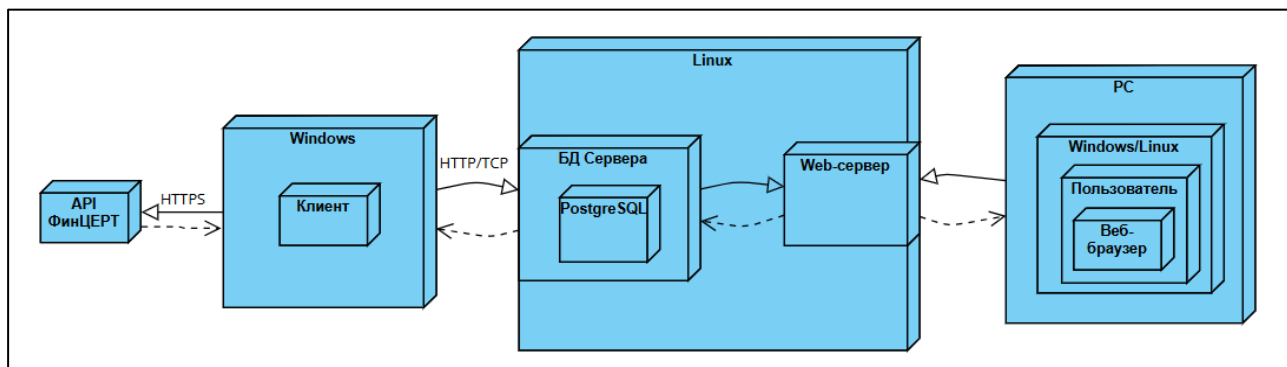


Рисунок 4 — UML-диаграмма развертывания

На рисунке представлено взаимодействие компонентов приложения в виде диаграммы 5 последовательности.

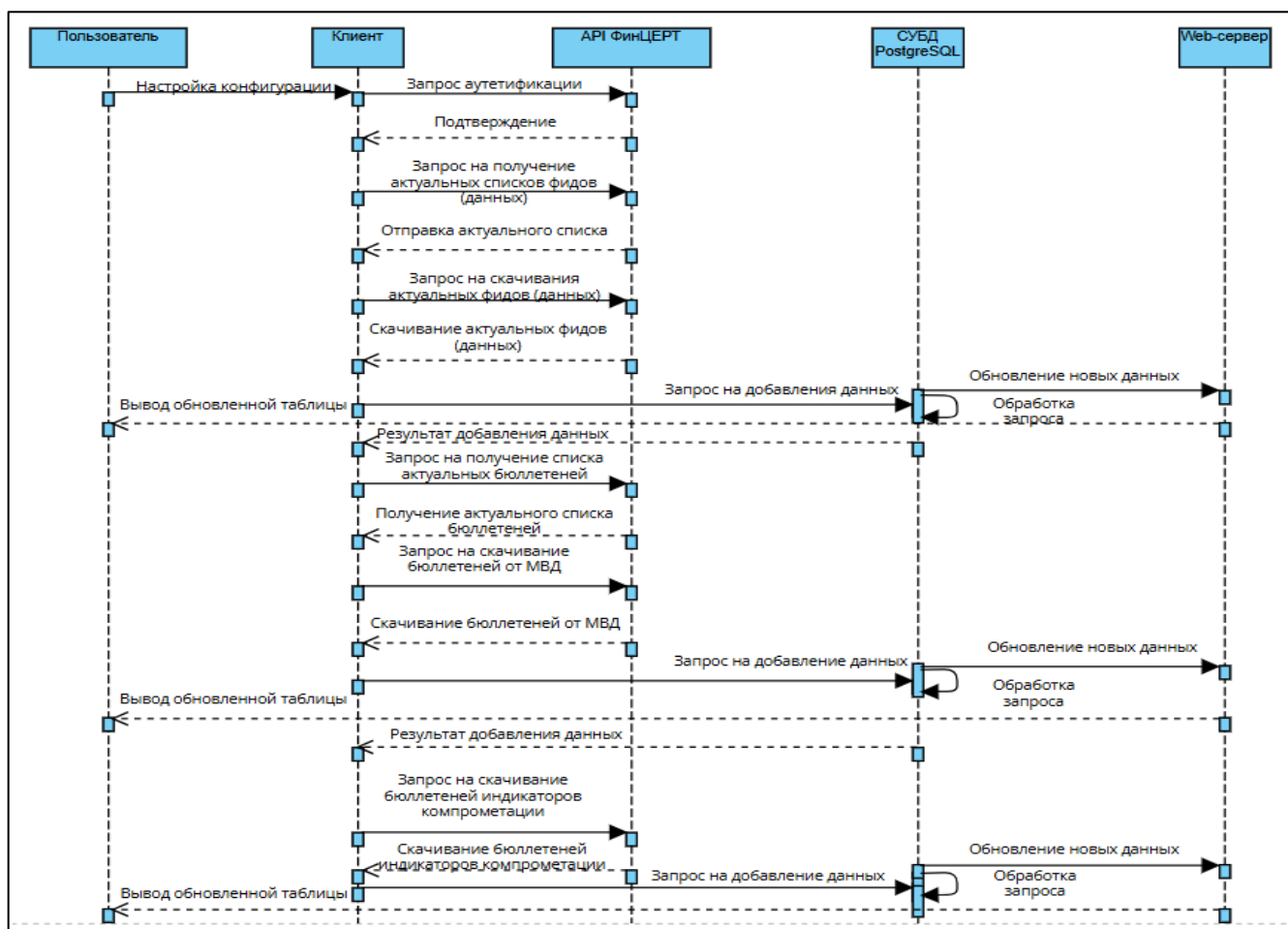


Рисунок 5 — UML-диаграмма последовательности

Диаграмма классов , представляющая модель данных, изображена на рисунке 6.

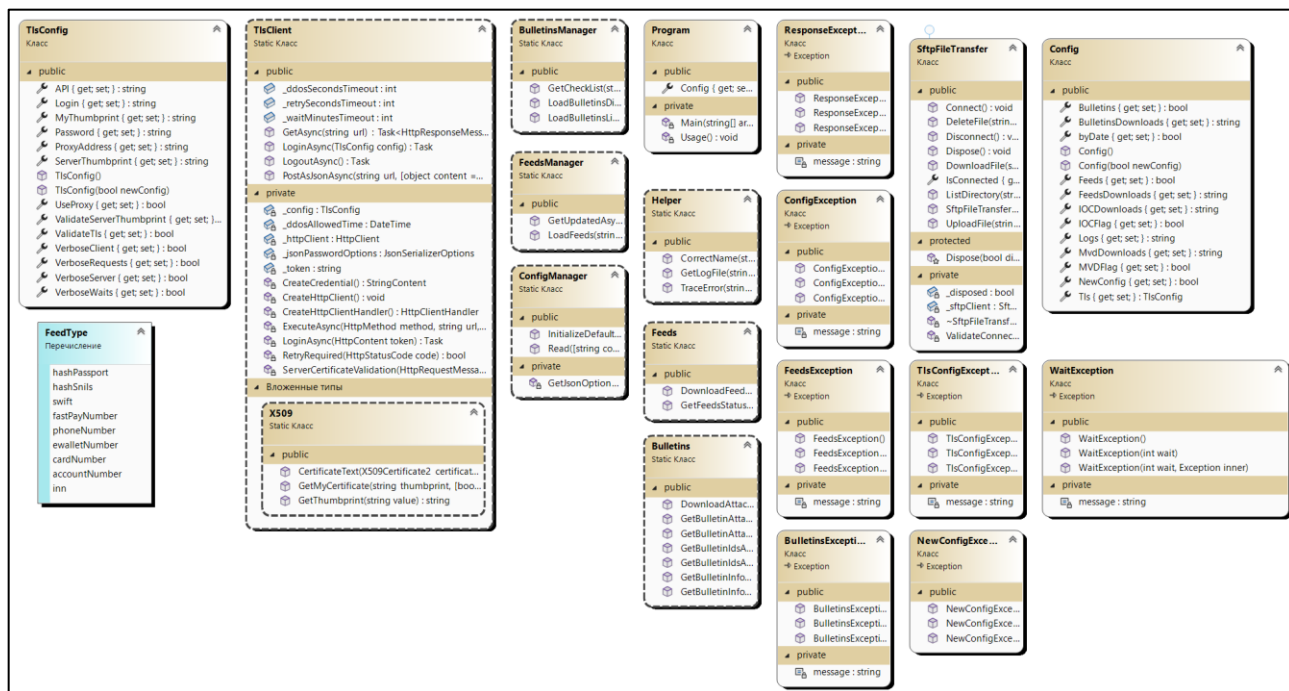


Рисунок 6 — UML-диаграмма классов

Диаграмма компонентов, изображена на рисунке 7.

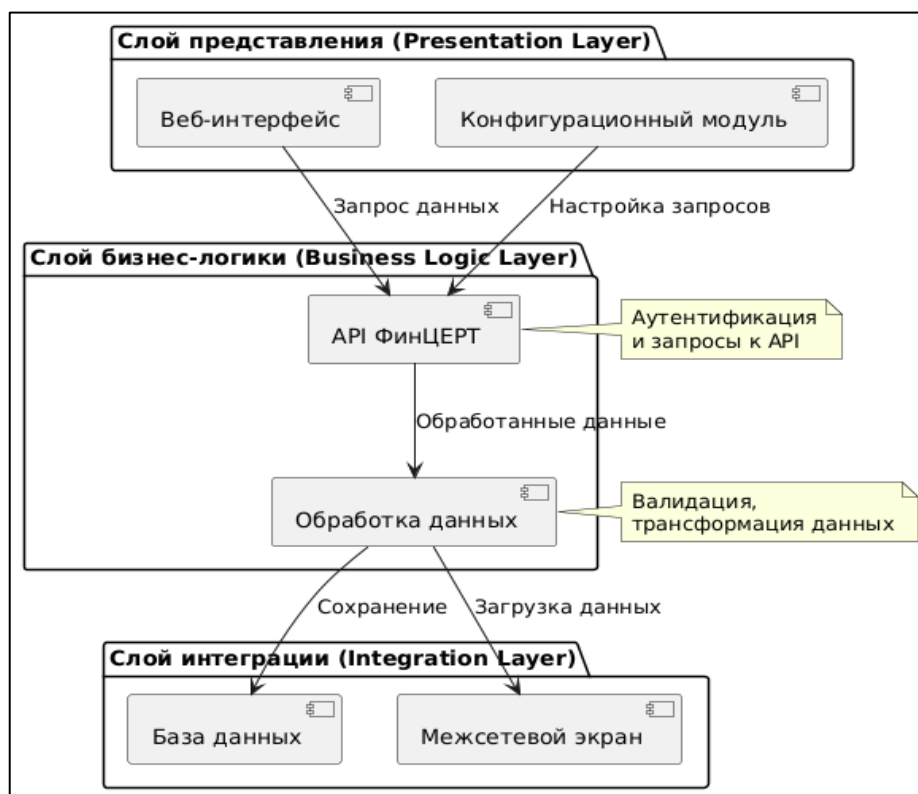


Рисунок 7 — UML-диаграмма компонентов

Диаграмма вариантов использования представлена на рисунке 8.

Изм.	Лист	№ докум.	Подпись	Дата

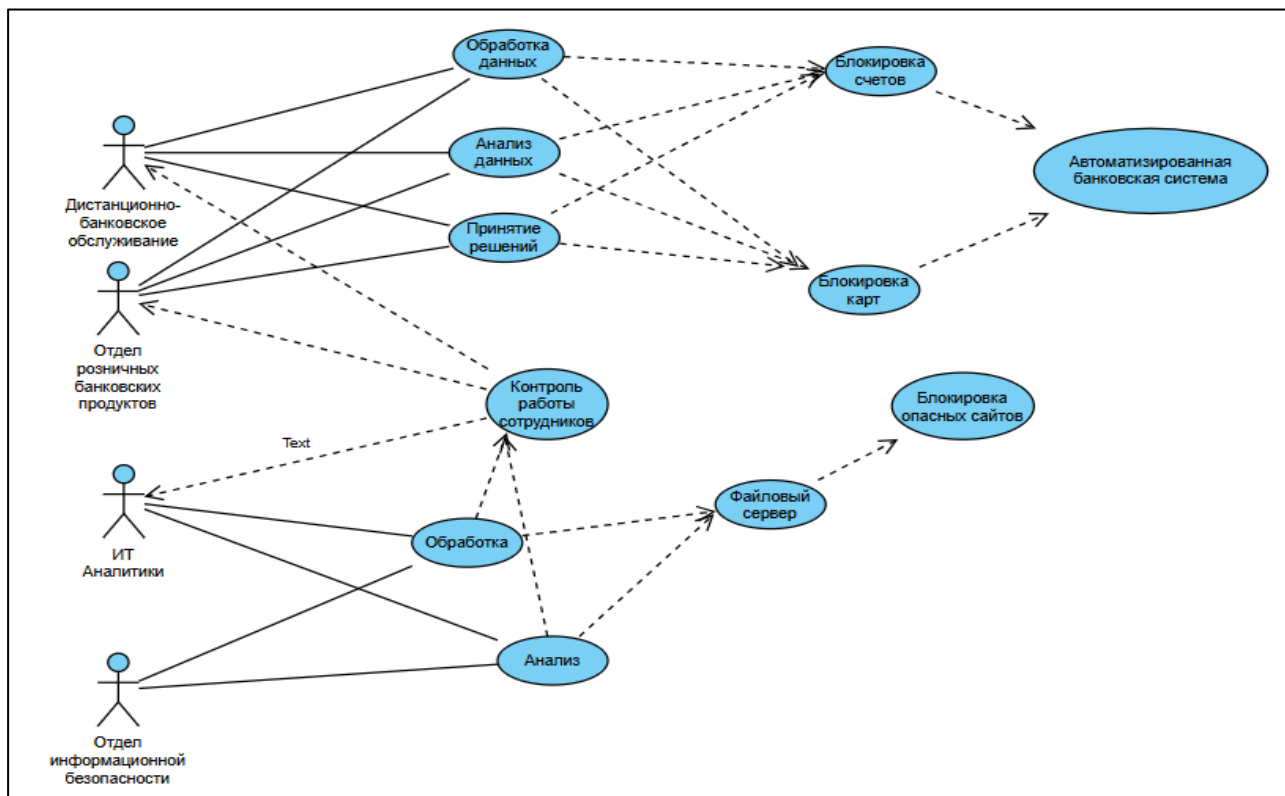


Рисунок 8 – UML-диаграмма вариантов использования

1.3.2 Проектирование базы данных

Для хранения обработанных данных, необходимо реализовать базу данных со следующими таблицами:

- FeedsTypeNavigation – таблица содержащая, название типов фидов (данных), и их описание (таблица 1);
- FeedsFincertDateDownloads – таблица хранящая, имя скачанных данных, дату загрузки (таблица 2);
- FeedsMvdDateDownloads — таблица содержащая, имя скачанных данных от МВД, дату загрузки (таблица 3);
- FincertIOCDateProcessing – таблица хранящая, дату скачивания и дату и время обработки данных (таблица 4);
- FeedsAccountNumbers – таблица, хранящая номера счетов, БИК, даты фиксации и количество мошеннических инцидентов граждан (таблица 5);
- FeedsCardNumbers – таблица содержащая номера карт, дату фиксации инцидента и их количество (таблица 6);

- FeedsEwalletNumbers – таблица хранящая, номера кошельков, название платежных систем, дату фиксации и их количество (таблица 7);
- FeedsFastpayNumbers – таблица хранящая, номера телефонов FastPay, дату фиксации и их количество (таблица 8);
- FeedsInn – таблица хранящая, ИНН, дату фиксации и количество инцидентов (таблица 9);
- FeedsPassportHash – таблица содержащая, хэш паспорта, дату фиксации и количество инцидентов (таблица 10);
- FeedsPhoneNumbers — таблица хранящая, номера телефонов, дату фиксации и их количество (таблица 11);
- FeedsSnilsHash – таблица хранящая, хэш СНИЛС, дату фиксации и количество инцидентов (таблица 12);
- FeedsSWIFT – таблица содержащая, номера счетов, SWIFT БИК, дату фиксации и количество инцидентов (таблица 13);
- FeedsMvdAccountNumbers – таблица хранящая, номера счетов от МВД и БИК (таблица 14);
- FeedsMvdFastpayNumbers – таблица хранящая, номера телефона Fastpay от МВД (таблица 15);
- FeedsMvdCardNumbers – таблица хранящая, номера карт от МВД (таблица 16);
- FeedsMvdInn – таблица хранящая, ИНН от МВД (таблица 17);
- FeedsMvdPassportHash – таблица содержащая, хэш паспорта от МВД (таблица 18);

Таблица 1 — Атрибуты таблицы FeedsTypeNavigation

Имя столбца	Тип данных	Описание	Роль атрибута	Пример
Id	SERIAL	Идентификатор типа данных	PK	1

Имя столбца	Тип данных	Описание	Роль атрибута	Пример
feeds_name	TEXT	Имя типа данных		МВД
feeds_description	TEXT	Описание типов данных		Типы МВД

Таблица 2 — Атрибуты таблицы FeedsFincertDateDownloads

Имя столбца	Тип данных	Описание	Роль атрибута	Пример
Id	SERIAL	Идентификатор полученных данных	PK	1
date	TEXT	Дата скачивания		2025-20-10
feedtype_id	INTEGER	Тип полученных данных	FK	1

Таблица 3 — Атрибуты таблицы FeedsMvdDateDownloads

Имя столбца	Тип данных	Описание	Роль атрибута	Пример
id	SERIAL	Идентификатор полученных данных	PK	1
date	DATE	Дата скачивания файлов		2025-20-01

Имя столбца	Тип данных	Описание	Роль атрибута	Пример
nameFeedHrid	TEXT	Название полученных данных		FinCERT- 20250320- 12334
feedtype_id	INTEGER	Тип полученных данных	FK	1

Таблица 4 — Атрибуты таблицы FincertIOCDateProcessing

Имя столбца	Тип данных	Описание	Роль атрибута	Пример
id	SERIAL	Идентификатор полученных данных	PK	1
dateMailing	DATE	Дата рассылки файлов		2025-01-20 15:30
iocHrid	TEXT	Название полученных данных		FinCERT- 20250320- 12334
feedstype_id	INTEGER	Тип полученных данных	FK	1
date	DATE	Дата обработки данных		2025-01-20

Таблица 5 — Атрибуты таблицы FeedsAccountNumbers

Имя столбца	Тип данных	Описание	Роль атрибута	Пример
id	SERIAL	Идентификатор полученных данных	PK	1
accountNumbers	TEXT	Номер лицевого счета		3213732934823 2
bic	TEXT	БИК		543423
dateFixed	DATE	Дата фиксации инцидента		2025-01-20
count	INTEGER	Количество инцидентов		3
country	TEXT	Страна		RUS
datedownloads_id	INTEGER	Идентификатор скачивания файлов	FK	1
dateExcluded	DATE	Дата исключения человека из списка		2025-01-20

Таблица 6 — Атрибуты таблицы FeedsCardNumbers

Имя столбца	Тип данных	Описание	Роль атрибута	Пример
id	SERIAL	Идентификатор полученных данных	PK	1

Имя столбца	Тип данных	Описание	Роль атрибута	Пример
cardNumbers	TEXT	Номера карт		2342232442233
dateFixed	DATE	Дата фиксации инцидента		2025-01-20
count	INTEGER	Количество инцидентов		3
country	TEXT	Страна		RUS
datedownloads_id	INTEGER	Идентификатор скачивания файлов	FK	1
dateExcluded	DATE	Дата исключения человека из списка		2025-01-20

Таблица 7 — Атрибуты таблицы FeedsEwalletNumbers

Имя столбца	Тип данных	Описание	Роль атрибута	Пример
id	SERIAL	Идентификатор полученных данных	PK	1
ewalletNumbers	TEXT	Номера кошельков		45238900323
paymentSystem	TEXT	Названия платежной системы		QIWI-Кошелек

Имя столбца	Тип данных	Описание	Роль атрибута	Пример
dateFixed	DATE	Дата фиксации инцидента		2025-01-20
count	INTEGER	Количество инцидентов		3
country	TEXT	Страна		RUS
datedownloads_id	INTEGER	Идентификатор скачивания файлов	FK	1
dateExcluded	DATE	Дата исключения человека из списка		2025-01-20

Таблица 8 — Атрибуты таблицы FeedsFastpayNumbers

Имя столбца	Тип данных	Описание	Роль атрибута	Пример
id	SERIAL	Идентификатор полученных данных	PK	1
FastpayNumbers	TEXT	Номера телефонов FastPay		79823898929
dateFixed	DATE	Дата фиксации инцидента		2025-01-20

Имя столбца	Тип данных	Описание	Роль атрибута	Пример
count	INTEGER	Количество инцидентов		3
country	TEXT	Страна		RUS
datedownloads_id	INTEGER	Идентификатор скачивания файлов	FK	1
dateExcluded	DATE	Дата исключения человека из списка		2025-01-20

Таблица 9 — Атрибуты таблицы FeedsInn

Имя столбца	Тип данных	Описание	Роль атрибута	Пример
id	SERIAL	Идентификатор полученных данных	PK	1
inn	TEXT	ИНН		678943847
dateFixed	DATE	Дата фиксации инцидента		2025-01-20
count	INTEGER	Количество инцидентов		3
country	TEXT	Страна		RUS

Имя столбца	Тип данных	Описание	Роль атрибута	Пример
datedownloads_id	INTEGER	Идентификатор скачивания файлов	FK	1
dateExcluded	DATE	Дата исключения человека из списка		2025-01-20

Таблица 10 — Атрибуты таблицы FeedsPassportHash

Имя столбца	Тип данных	Описание	Роль атрибута	Пример
id	SERIAL	Идентификатор полученных данных	PK	1
passportHash	TEXT	Хэш паспортов		D@as53wqd23 31h21ncsn3%2 nlwdk343dna43 23123nasl1@
dateFixed	DATE	Дата фиксации инцидента		2025-01-20
count	INTEGER	Количество инцидентов		3

Имя столбца	Тип данных	Описание	Роль атрибута	Пример
country	TEXT	Страна		RUS
datedownloads_ id	INTEGER	Идентификатор скачивания файлов	FK	1
dateExcluded	DATE	Дата исключения человека из списка		2025-01-20

Таблица 11 — Атрибуты таблицы FeedsPhoneNumbers

Имя столбца	Тип данных	Описание	Роль атрибута	Пример
id	SERIAL	Идентификатор полученных данных	PK	1
phoneNumbers	TEXT	Номера телефонов		79823898929
dateFixed	DATE	Дата фиксации инцидента		2025-01-20
count	INTEGER	Количество инцидентов		3
country	TEXT	Страна		RUS
datedownloads_ id	INTEGER	Идентификатор скачивания файлов	FK	1

Имя столбца	Тип данных	Описание	Роль атрибута	Пример
dateExcluded	DATE	Дата исключения человека из списка		2025-01-20

Таблица 12 — Атрибуты таблицы FeedsSnilsHash

Имя столбца	Тип данных	Описание	Роль атрибута	Пример
id	SERIAL	Идентификатор полученных данных	PK	1
hashSnils	TEXT	Хэш СНИЛС		D@as53wqd23 31h21ncsn3%2 nlwdk343dna43 23123nasl1@
dateFixed	DATE	Дата фиксации инцидента		2025-01-20
count	INTEGER	Количество инцидентов		3
country	TEXT	Страна		RUS
datedownloads_ id	INTEGER	Идентификатор скачивания файлов	FK	1
dateExcluded	DATE	Дата исключения человека из списка		2025-01-20

Таблица 13 — Атрибуты таблицы FeedsSWIFT

Имя столбца	Тип данных	Описание	Роль атрибута	Пример
id	SERIAL	Идентификатор полученных данных	PK	1
accountNumbers- Swift	TEXT	Номер счета SWIFT		2617231932138 12382
swiftBIC	TEXT	SWIFT БИК		72138761238
dateFixed	DATE	Дата фиксации инцидента		2025-01-20
count	INTEGER	Количество инцидентов		3
country	TEXT	Страна		RUS
datedownloads_id	INTEGER	Идентификатор скачивания файлов	FK	1
dateExcluded	DATE	Дата исключения человека из списка		2025-01-20

Таблица 14 — Атрибуты таблицы FeedsMvdAccountNumbers

Имя столбца	Тип данных	Описание	Роль атрибута	Пример
id	SERIAL	Идентификатор полученных данных от МВД	PK	1

Имя столбца	Тип данных	Описание	Роль атрибута	Пример
accountNumbers	TEXT	Номер лицевого счета от МВД		3213732934823 2
bic	TEXT	БИК от МВД		543423
datedownloads_id	INTEGER	Идентификатор скачивания файлов от МВД	FK	1
dateExcluded	DATE	Дата исключения человека из списка		2025-01-20

Таблица 15 — Атрибуты таблицы FeedsMvdFastpayNumbers

Имя столбца	Тип данных	Описание	Роль атрибута	Пример
id	SERIAL	Идентификатор полученных данных от МВД	PK	1
FastpayNumbers	TEXT	Номера телефонов FastPay от МВД		79823898929
datedownloads_id	INTEGER	Идентификатор скачивания файлов от МВД	FK	1

Имя столбца	Тип данных	Описание	Роль атрибута	Пример
dateExcluded	DATE	Дата исключения человека из списка		2025-01-20

Таблица 16 — Атрибуты таблицы FeedsMvdCardNumbers

Имя столбца	Тип данных	Описание	Роль атрибута	Пример
id	SERIAL	Идентификатор полученных данных от МВД	PK	1
cardNumbers	TEXT	Номера карт от МВД		9828828228329 992
datedownloads_id	INTEGER	Идентификатор скачивания файлов от МВД	FK	1
dateExcluded	DATE	Дата исключения человека из списка		2025-01-20

Таблица 17 — Атрибуты таблицы FeedsMvdInn

Имя столбца	Тип данных	Описание	Роль атрибута	Пример
id	SERIAL	Идентификатор полученных данных от МВД	PK	1

Имя столбца	Тип данных	Описание	Роль атрибута	Пример
inn	TEXT	ИНН от МВД		678943847
datedownloads_id	INTEGER	Идентификатор скачивания файлов от МВД	FK	1
dateExcluded	DATE	Дата исключения человека из списка		2025-01-20

Таблица 18 — Атрибуты таблицы FeedsMvdPassportHash

Имя столбца	Тип данных	Описание	Роль атрибута	Пример
id	SERIAL	Идентификатор полученных данных от МВД	PK	1
passportHash	TEXT	Хэш паспортов от МВД		D@as53wqd23 31h21ncsn3%2 nlwdk343dna43 23123nasl1@
datedownloads_id	INTEGER	Идентификатор скачивания файлов от МВД	FK	1


```
CREATE DATABASE IF NOT EXISTS users;
```

Затем создадим внутри БД необходимые таблицы. В данном разделе будет описан процесс создания и работа с основными таблицами БД.

Для создания таблицы, которая будет служить навигацией для остальных таблиц, напомним следующий запрос:

```
CREATE TABLE feeds_type_navigation (  
    id SERIAL PRIMARY KEY,  
    feeds_name VARCHAR(20) NOT NULL,  
    feeds_description TEXT NOT NULL  
);
```

Для создания таблицы, которая будет хранить дату загрузки последних скачиваемых фидов (данных), напомним следующий запрос:

```
CREATE TABLE feeds_fincert_date_downloads (  
    id SERIAL PRIMARY KEY,  
    feedstype_id INTEGER NOT NULL REFERENCES  
    feeds_type_navigation(id) ON DELETE CASCADE,  
    date VARCHAR(30) NOT NULL  
);
```

Для создания таблицы, которая будет хранить название и дату скачиваемых фидов (данных) от МВД, напомним следующий запрос:

```
CREATE TABLE feeds_mvd_date_downloads (  
    id SERIAL PRIMARY KEY,  
    feedstype_id INTEGER NOT NULL REFERENCES  
    feeds_type_navigation(id) ON DELETE CASCADE,  
    date VARCHAR(30) NOT NULL,  
    nameFeedsTrim VARCHAR(50) NOT NULL  
);
```

Для создания таблицы, которая будет хранить дату рассылки, дату обработки и название скачиваемых данных, напомним следующий запрос:

```
CREATE TABLE fin_cert_ioc_date_processing (  
    id SERIAL PRIMARY KEY,  
    feedstype_id INTEGER NOT NULL REFERENCES  
    feeds_type_navigation(id) ON DELETE CASCADE,
```

```

        dateMailing VARCHAR(30) NOT NULL,
        date VARCHAR(30),
        iocTrim VARCHAR(50) NOT NULL
    );

```

Для создания таблицы, которая будет хранить содержимое скачиваемых файлов с номерами счетов и БИК неблагонадежных клиентов:

```

CREATE TABLE feeds_account_numbers (
    id SERIAL PRIMARY KEY,
    datedownloads_id INTEGER REFERENCES
    feeds_fincert_date_downloads(id) ON DELETE SET NULL,
    accountNumbers VARCHAR(64) NOT NULL,
    bic VARCHAR(64) NOT NULL,
    dateFixed VARCHAR(30) NOT NULL,
    count SMALLINT NOT NULL,
    country VARCHAR(15) NOT NULL,
    dateExcluded TIMESTAMP
);

```

Для создания таблицы, которая будет хранить содержимое скачиваемых файлов с номерами карт неблагонадежных клиентов:

```

CREATE TABLE feeds_card_numbers (
    id SERIAL PRIMARY KEY,
    datedownloads_id INTEGER REFERENCES
    feeds_fincert_date_downloads(id) ON DELETE SET NULL,
    cardNumbers VARCHAR(64) NOT NULL,
    dateFixed VARCHAR(30) NOT NULL,
    count SMALLINT NOT NULL,
    country VARCHAR(15) NOT NULL,
    dateExcluded TIMESTAMP);

```

Для создания таблицы, которая будет хранить содержимое скачиваемых файлов с номерами электронных кошельков неблагонадежных клиентов, напишем следующий запрос:

```

CREATE TABLE feeds_ewallet_numbers (
    id SERIAL PRIMARY KEY,

```

```

        datedownloads_id INTEGER REFERENCES
feeds_fin_cert_date_downloads(id) ON DELETE SET NULL,
        ewalletNumbers VARCHAR(64) NOT NULL,
        paymentSystem VARCHAR(100) NOT NULL,
        dateFixed VARCHAR(30) NOT NULL,
        count SMALLINT NOT NULL,
        country VARCHAR(15) NOT NULL,
        dateExcluded TIMESTAMP
    );

```

Для создания таблицы которая будет хранить содержимое скачиваемых файлов с номерами телефонов для быстрых платежей неблагонадежных клиентов, напомним следующий запрос:

```

CREATE TABLE feeds_fastpay_numbers (
    id SERIAL PRIMARY KEY,
    datedownloads_id INTEGER REFERENCES
feeds_fin_cert_date_downloads(id) ON DELETE SET NULL,
    fastpayNumber VARCHAR(100) NOT NULL,
    dateFixed VARCHAR(30) NOT NULL,
    count SMALLINT NOT NULL,
    country VARCHAR(15) NOT NULL,
    dateExcluded TIMESTAMP
);

```

Для создания таблицы которая будет хранить содержимое скачиваемых файлов с ИНН неблагонадежных клиентов, напомним следующий запрос:

```

CREATE TABLE feeds_inn (
    id SERIAL PRIMARY KEY,
    datedownloads_id INTEGER REFERENCES
feeds_fin_cert_date_downloads(id) ON DELETE SET NULL,
    inn VARCHAR(30) NOT NULL,
    dateFixed VARCHAR(30) NOT NULL,
    count SMALLINT NOT NULL,
    country VARCHAR(15) NOT NULL,
    dateExcluded TIMESTAMP
);

```

Для создания таблицы, которая будет хранить содержимое скачиваемых файлов с хэшем паспортов неблагонадежных клиентов, напомним следующий запрос:

```
CREATE TABLE feeds_passport_hash (
    id SERIAL PRIMARY KEY,
    datedownloads_id INTEGER REFERENCES
    feeds_fin_cert_date_downloads(id) ON DELETE SET NULL,
    passportHash VARCHAR(150) NOT NULL,
    dateFixed VARCHAR(30) NOT NULL,
    count SMALLINT NOT NULL,
    country VARCHAR(15) NOT NULL,
    dateExcluded TIMESTAMP
);
```

Для создания таблицы, которая будет хранить содержимое скачиваемых файлов с номерами телефонов неблагонадежных клиентов, напомним следующий запрос:

```
CREATE TABLE feeds_phone_numbers (
    id SERIAL PRIMARY KEY,
    datedownloads_id INTEGER REFERENCES
    feeds_fincert_date_downloads(id) ON DELETE SET NULL,
    phoneNumbers VARCHAR(64) NOT NULL,
    dateFixed VARCHAR(30) NOT NULL,
    count SMALLINT NOT NULL,
    country VARCHAR(15) NOT NULL,
    dateExcluded TIMESTAMP
);
```

Для создания таблицы, которая будет хранить содержимое скачиваемых файлов с хэшами СНИЛС неблагонадежных клиентов, напомним следующий запрос:

```
CREATE TABLE feeds_snils_hash (
    id SERIAL PRIMARY KEY,
    datedownloads_id INTEGER REFERENCES
    feeds_fincert_date_downloads(id) ON DELETE SET NULL,
    hashSnils VARCHAR(200) NOT NULL,
```

```

        dateFixed VARCHAR(30) NOT NULL,
        count SMALLINT NOT NULL,
        country VARCHAR(15) NOT NULL,
        dateExcluded TIMESTAMP
    );

```

Для создания таблицы, которая будет хранить содержимое скачиваемых файлов с реквизитами SWIFT неблагонадежных клиентов, напомним следующий запрос:

```

CREATE TABLE feeds_swift (
    id SERIAL PRIMARY KEY,
    datedownloads_id INTEGER REFERENCES
    feeds_fincert_date_downloads(id) ON DELETE SET NULL,
    accountNumbersSwift VARCHAR(150) NOT NULL,
    swiftBIC VARCHAR(50) NOT NULL,
    dateFixed VARCHAR(30) NOT NULL,
    count SMALLINT NOT NULL,
    country VARCHAR(15) NOT NULL,
    dateExcluded TIMESTAMP
);

```

Для создания таблицы, которая будет хранить содержимое скачиваемых файлов с номерами счетов от МВД неблагонадежных клиентов, напомним следующий запрос:

```

CREATE TABLE feeds_mvd_account_numbers (
    id SERIAL PRIMARY KEY,
    datedownloads_id INTEGER REFERENCES
    feeds_mvd_date_downloads(id) ON DELETE SET NULL,
    accountNumbers VARCHAR(64) NOT NULL,
    bic VARCHAR(64) NOT NULL,
    dateExcluded TIMESTAMP
);

```

Для создания таблицы, которая будет хранить содержимое скачиваемых файлов с номерами карт от МВД неблагонадежных клиентов, напомним следующий запрос:

```

CREATE TABLE feeds_mvd_card_numbers (

```

```

        id SERIAL PRIMARY KEY,
        datedownloads_id INTEGER REFERENCES
        feeds_mvd_date_downloads(id) ON DELETE SET NULL,
        cardNumbers VARCHAR(64) NOT NULL,
        dateExcluded TIMESTAMP
    );

```

Для создания таблицы, которая будет хранить содержимое скачиваемых файлов с номерами телефонов для быстрых платежей от МВД неблагонадежных клиентов, напишем следующий запрос:

```

CREATE TABLE feeds_mvd_fast_pay_numbers (
    id SERIAL PRIMARY KEY,
    datedownloads_id INTEGER REFERENCES
    feeds_mvd_date_downloads(id) ON DELETE SET NULL,
    fastpayNumber VARCHAR(64) NOT NULL,
    dateExcluded TIMESTAMP
);

```

Для создания таблицы, которая будет хранить содержимое скачиваемых файлов с ИНН от МВД неблагонадежных клиентов, напишем следующий запрос:

```

CREATE TABLE feeds_mvd_inn (
    id SERIAL PRIMARY KEY,
    datedownloads_id INTEGER REFERENCES
    feeds_mvd_date_downloads(id) ON DELETE SET NULL,
    inn VARCHAR(30) NOT NULL,
    dateExcluded TIMESTAMP
);

```

Для создания таблицы, которая будет хранить содержимое скачиваемых файлов с хэшами паспортов от МВД неблагонадежных клиентов, напишем следующий запрос:

```

CREATE TABLE feeds_mvd_passport_hash (
    id SERIAL PRIMARY KEY,
    datedownloads_id INTEGER REFERENCES
    feeds_mvd_date_downloads(id) ON DELETE SET NULL,
    passportHash VARCHAR(150) NOT NULL,

```

```
dateExcluded TIMESTAMP
```

```
);
```

Для добавления данных в таблицу использовалась временная таблица, которая будет удалена после использования, пример создания временной таблицы:

```
CREATE TEMP TABLE public_tempusersaccountNumbers  
(accountNumbers, bic, dateFixed, count, country);
```

Для проверки на наличие данных в основной таблице, заполняем временную таблицу скаченными файлами из CSV файла

```
COPY public_tempusersaccountNumbers(accountNumbers, bic,  
dateFixed, count, country)  
FROM "C:\feeds\account_numbers.csv"  
DELIMITER ','  
CSV HEADER  
ENCODING 'UTF-8';
```

Для добавления скачиваемых данных в основную таблицу, сравниваем ее с временной, и данных, которых нет в основной таблице, добавляем, пример запроса для одной таблицы:

```
INSERT INTO  
public.users_feedsfincertaccountnumbers(accountNumbers, bic,  
count, country))  
SELECT s.accountNumbers  
FROM public.tempusers_ feedsfincertaccountnumbers s  
WHERE NOT EXISTS(  
    SELECT 1  
    FROM public.users_feedsfincertaccountnumbers t  
    WHERE t.accountNumbers AND s.accountNumbers  
);
```

Для добавления в столбец даты для исключенных клиентов, сравнивается временная и основная таблица, данные которые есть в основной таблице, но нет в временной помечаем, как исключенные из списка и вносим текущую дату в столбец, пример запроса для одной таблицы:

```
UPDATE public.users_feedsfincertaccountnumbers t
```

					ПГУ 1.090301.09.001 ПЗ	Лист
						42
Изм.	Лист	№ докум.	Подпись	Дата		

```

SET dateExcluded = CURRENT_DATE
WHERE NOT EXISTS (
    SELECT 1
    FROM public.tempusers_feedsfincertaccountnumbers s
    WHERE t.accountNumbers AND s.accountNumbers)
AND t.dateExcluded is NULL;

```

Для удаления временной таблицы после ее использования, напишем следующий запрос:

```
DROP TABLE IF EXISTS {tempTableName};
```

1.4.2 Описание программы

В папке logs/feeds содержатся текстовые файлы, содержащие сведения об ошибках, об работе программы. Данные файлы необходимы для отслеживания работы программы и своевременное реагирование на ошибки и быстрое их устранения.

В папке logs/ShedulerTask содержатся текстовые файлы, который содержат информацию о работе службы, в данных файлах, прописывается время начала работы службы, код возврата, время завершения службы и отслеживает появление новых файлов в папках.

Файл Config.cs представляет из себя класс для хранения настроек приложения. Данный класс предназначен для работы с файлами конфигурации приложения.

```

using TLS;
namespace FincertClient;
public class Config
{
    public bool NewConfig { get; set; }
    public TlsConfig? Tls { get; set; }
    public bool Feeds { get; set; }
    public string? FeedsDownloads { get; set; }
    public bool Bulletins { get; set; }
    public string? BulletinsDownloads { get; set; }
    public bool byDate { get; set; }
    public string? MvdDownloads { get; set; }
}

```

```

public string ? IOCDownloads { get; set; }
public string? Logs { get; set; }
public bool MVDFlag { get; set; }
public bool IOCFlag { get; set; }
public Config() { }
public Config(bool newConfig)
{
    NewConfig = newConfig;
    Tls = new(newConfig);
    Feeds = false;
    FeedsDownloads = nameof(FeedsDownloads);
    byDate = true;
    Bulletins = false;
    BulletinsDownloads = nameof(BulletinsDownloads);
    MvdDownloads = nameof(MvdDownloads);
    IOCDownloads = nameof(IOCDownloads);
    Logs = nameof(Logs);
    MVDFlag = false;
    IOCFlag = false;
}
}

```

Класс ConfigManager.cs является менеджером конфигурации для приложения, который управляет загрузкой, созданием и валидацией JSON-конфигурационных файлов .

```

using System.Text.Json;
namespace FincertClient.Managers;
internal static class ConfigManager
{
    private static JsonSerializerOptions GetJsonOptions()
    {
        return new()
        {
            WriteIndented = true
        };
    }
}

```

```

public static Config Read(string configType = null)
{
    string configFileNames = configType switch
    {
        "dailyMVD" => "MVD.config.json",
        "dailyIOC" => "IOC.config.json",
        "FinCERTFeeds"=>"FinCERT-Client.config.json",
        _ => "Нет Конфига"
    };
    Console.WriteLine(configFileName);
    string appsettings =
Path.Combine(AppContext.BaseDirectory, configFileNames);
    if (File.Exists(appsettings))
    {
        using var read = File.OpenRead(appsettings);
        var config = JsonSerializer.Deserialize<Config>(read);
        if (config is null)
            throw new NewConfigurationException(
                @"$Файл настроек "{appsettings}" испорчен -
удалите его.");
        if (config.NewConfig)
            throw new NewConfigurationException(
                @"$Откорректируйте файл настроек
"{appsettings}" и отключите NewConfig.");
        return config;
    }
    var newConfig = new Config(true);
    using var write = File.OpenWrite(appsettings);
    JsonSerializer.Serialize(write, newConfig,
GetJsonOptions());
    throw new NewConfigurationException(
        @"$Создан новый файл настроек "{appsettings}" -
откорректируйте его.");
}

public static void InitializeDefaultConfigs()

```

					ПГУ 1.090301.09.001 ПЗ	Лист
						45
Изм.	Лист	№ докум.	Подпись	Дата		

```

{
    string[] configTypes = { "MVD", "IOC", "FinCERTFeeds" };

    foreach (var type in configTypes)
    {
        string fileName =
Path.Combine(AppContext.BaseDirectory, $"{type}.config.json");
        if (!File.Exists(fileName))
        {
            var newConfig = new Config(true);
            using var write = File.OpenWrite(fileName);
            JsonSerializer.Serialize(write, newConfig,
GetJsonOptions());
        }
    }
}

internal class NewConfigException : Exception
{
    public NewConfigException()
        : base() { }

    public NewConfigException(string message)
        : base(message) { }

    public NewConfigException(string message, Exception inner)
        : base(message, inner) { }
}

public class ConfigException : Exception
{
    const string message = "Параметр '{0}' не указан в конфиге.";
    public ConfigException()
        : base() { }

    public ConfigException(string paramName)
        : base(string.Format(message, paramName)) { }

    public ConfigException(string paramName, Exception inner)

```

```

        : base(string.Format(message, paramName), inner) { }
    }

```

Класс `TLSClient.cs` – это статический класс, реализующий безопасное TLS-подключение к HTTP-серверу с использованием клиентских сертификатов.

Основные поля класса:

```

private static HttpClient _httpClient = null!;
private static TlsConfig _config = null!;
private static string _token = string.Empty;
private static readonly JsonSerializerOptions
    _jsonPasswordOptions = new()
{ Encoder = JavaScriptEncoder.UnsafeRelaxedJsonEscaping };

private static DateTime _ddosAllowedTime = DateTime.Now;
public static readonly int _retrySecondsTimeout = 2;
public static readonly int _ddosSecondsTimeout = 1;
public static readonly int _waitMinutesTimeout = 10;

```

Метод `LoginAsync(TlsConfig config)` является основным методом для подключения к серверу.

```

public static async Task LoginAsync(TlsConfig config)
{
    _config = config;
    var token = CreateCredential();
    CreateHttpClient();
    await LoginAsync(token);
}

```

Метод `CreateCredential()` создает JSON-содержимое для аутентификации. Метод возвращает строку с логином и паролем в формате JSON.

```

private static StringContent CreateCredential()
{
    string api = _config.API!;
    string login = _config.Login!;
    string password = _config.Password!;
    var cred = new Credential(login, password);
}

```

```

        string json = JsonSerializer.Serialize(cred,
_jsonPasswordOptions);
        var token = new StringContent(json, Encoding.UTF8,
"application/json");
        return token;}

```

Метод `CreateHttpClientHandler()` создает и настраивает обработчик HTTP-запросов. Метод настраивает параметры TLS, добавляет клиентский сертификат, настраивает прокси (если требуется).

```

        private static HttpClientHandler CreateHttpClientHandler()
    {
        HttpClientHandler handler = new()
        {
            UseDefaultCredentials = false,
            ClientCertificateOptions =
ClientCertificateOption.Manual,
            ServerCertificateCustomValidationCallback =
ServerCertificateValidation,
            SslProtocols = SslProtocols.Tls12
        };
        string thumbprint =
X509.GetThumbprint(_config.MyThumbprint!);
        X509Certificate2 certificate =
X509.GetMyCertificate(thumbprint);
        handler.ClientCertificates.Add(certificate);
        return handler;
    }

```

Метод `CreateHttpClient()` инициализирует HTTP-клиент с настройками.

```

        private static void CreateHttpClient()
    {
        var handler = CreateHttpClientHandler();
        _httpClient = new(handler, true);
        string app =
Path.GetFileNameWithoutExtension(Environment.ProcessPath)
        ?? nameof(TlsClient);
    }

```

```

string ver =
Assembly.GetEntryAssembly()?.GetName().Version?.ToString()
?? "8.0";
_httpClient.DefaultRequestHeaders.UserAgent.Add(
    new ProductInfoHeaderValue(app, ver));
MediaTypeWithQualityHeaderValue("application/json"));
}

```

Метод `LoginAsync (HttpContent token)` выполняет аутентификацию на сервере. Метод отправляет POST-запрос с учетными данными, сохраняет полученный токен.

```

private static async Task LoginAsync(HttpContent token)
{
    var response = await ExecuteAsync(HttpMethod.Post,
"account/login", token);
    // eyJhbGciOiJCJ9...eyJodHRwOi8
    _token = await response.Content.ReadAsStringAsync();
    // Authorization: Bearer eyJhbGciOiJCJ9...eyJodHRwOi8
    _httpClient.DefaultRequestHeaders.Authorization =
new("Bearer", _token);
}

```

Метод `LogoutAsync()` завершает сеанс работы. Он отправляет запрос на выход, очищает токен и освобождает ресурсы `HttpClient`.

```

public static async Task LogoutAsync()
{
    await ExecuteAsync(HttpMethod.Post, "account/logout");
    _token = string.Empty;
    _httpClient.DefaultRequestHeaders.Authorization =
new("Bearer", _token);
    _httpClient.Dispose();
}

```

Метод `PostAsJsonAsync(string url, object? Content)` выполняет POST-запрос с JSON-содержимым.

```

public static async Task<HttpResponseMessage>
PostAsJsonAsync(

```

```

        string url, object? content = null)
        => await ExecuteAsync(HttpMethod.Post, url,
JsonContent.Create(content));

```

Метод GetAsync(string url) выполняет GET-запрос.

```

        public static async Task<HttpResponseMessage>
GetAsync(string url)
        => await ExecuteAsync(HttpMethod.Get, url);

```

Метод ExecuteAsync() отвечает за выполнение запросов. Он формирует полный URL, обрабатывает защиту от DDoS, выполняет запрос с повторами при ошибках, анализирует коды ответа.

```

        private static async Task<HttpResponseMessage> ExecuteAsync(
HttpMethod method, string url, HttpContent? content = null)
        {
            int retry = 0;
            string api = _config.API + url;
            HttpStatusCode code = HttpStatusCode.Continue;
            DateTime end = DateTime.Now.AddMinutes(_waitMinutesTimeout);
            Trace.WriteLineIf(_config.VerboseRequests,
                $"{method} {url}"); // url | api
            while (DateTime.Now < end)
            {
                if (DateTime.Now < _ddosAllowedTime)
                    Thread.Sleep(_ddosAllowedTime - DateTime.Now);
                using (var request = new HttpRequestMessage(method, api))
                {
                    if (content != null)
                        request.Content = content;
                    retry++;
                    var response = await _httpClient.SendAsync(request);
                    _ddosAllowedTime =
DateTime.Now.AddSeconds(_ddosSecondsTimeout);
                    code = response.StatusCode;
                    if (!RetryRequired(code))
                    {
                        if (code == HttpStatusCode.OK)

```

```

        return response;
    else
        throw new ResponseException(code);
    }
}

int pause = retry * _retrySecondsTimeout;
Trace.WriteLineIf(_config.VerboseWaits,
    $"Код {code} - повтор {retry} через {pause} сек.");
Thread.Sleep(pause * 1000);
}

throw new WaitException(_waitMinutesTimeout);
}

```

Класс **BulletinManager.cs** отвечает за управление бюллетенями в приложении. Класс организует загрузку бюллетеней, обработку полученных данных, взаимодействует с базой данных.

Метод **LoadBulletinsDirs()** загружает список бюллетеней.

```

public static async Task LoadBulletinsDirs(string path, bool
MVDFlag, bool IOCFlag, int limit = 100, long offset = 0)
{
    string lastMvdZip = "feeds_20240101-01.zip";
    string lastIocZip = "FinCERT_20240101-01.zip";
    string currentDate = DateTime.Now.ToString("yyyyMMdd");
    var logger = new Logger(@"C:\logs");
    var cidrProcessor = new CidrProcessor(logger);
    var fileManager = new FileManager(logger,
@"C:\IOCDownloads\version.txt");
    var blockListProcessor = new IpBlockListProcessor(logger,
fileManager, cidrProcessor);
    var dataProcessor = new IpDataProcessor(logger);
    var processor = new FinCertProcessor();
    var ipChecker = new IpChecker(logger, fileManager,
blockListProcessor, dataProcessor, cidrProcessor);
    Directory.CreateDirectory(path);
    Trace.WriteLine("Получение бюллетеней по папкам...");
}

```

```

var ids = await GetBulletinIdsAsync(limit, offset);
Trace.WriteLine($"Всего: {ids.Total}, в листе:
{ids.Items.Length}.");
Trace.WriteLine($"Пропущен бюллетень {bulletin.Hrid} - не
соответствует текущей дате");
    continue;
}
var date = DateTime.Parse(bulletin!.PublishedDate);
var name = CorrectName($"{bulletin.Hrid.Trim()}");
var dir = Path.Combine(path, name);
StringBuilder sb = new();
sb.AppendLine(id)
    .AppendLine($"Дата публикации: {date:g}")
    .AppendLine($"Идентификатор: {bulletin.Hrid}")
    .AppendLine($"Заголовок: {bulletin.Header}")
    .AppendLine($"Тип рассылки: {bulletin.Type}")
    .AppendLine($"Подтип рассылки: {bulletin.Subtype}");
if (bulletin.Attachment != null)
{
    string filename = bulletin.Attachment.Name;
    sb.AppendLine($"Файл рассылки: {filename}");
    string attach = Path.Combine(dir, filename)
    if (filename.StartsWith("feeds_20",
StringComparison.OrdinalIgnoreCase) && MVDFlag == true)
    {
        Trace.WriteLine(name);
        Directory.CreateDirectory(dir);
        await
DownloadAttachmentAsync(bulletin.Attachment.Id, attach);
if (string.Compare(filename, lastMvdZip, true) > 0)
        {
Trace.WriteLine($"Получен фид МВД '{filename}'");
lastMvdZip = filename;
string dir1 = Program.Config.MvdDownloads!; //TODO
if (!dir1.Equals(string.Empty))

```

```

Directory.CreateDirectory(dir1);
ZipFile.ExtractToDirectory(attach, dir1, true);

var connector = new
PGSQLConnector("Host=localhost;Database=users;Username=postgres;Pa
ssword=shkola77dan");

connector.LoadMvdTypeTable(bulletin.Hrid, currentDate);
    }
}

if (filename.StartsWith("FinCERT_20",
StringComparison.OrdinalIgnoreCase) && IOCFlag == true)
{
    if (Directory.Exists(dir))
    {
        if (offset == 0)
        {
            Trace.WriteLine($"{name}    конец скачивания.");
            break;
        }
        else
        {
            Trace.WriteLine($"{name} // есть");
            continue;
        }
    }
}

```

Класс FeedsManager.cs представляет собой менеджер для работы с фидами (данными) из системы АСОИ ФинЦЕРТ. Он содержит методы для получения информации об обновлениях фидов и их загрузки.

```

using System.Diagnostics;
using API;
using static API.Feeds;
using FincertClient.DatebaseManager;
namespace FincertClient.Managers;
internal static class FeedsManager
{

```

```

        public static async Task<DateTime> GetUpdatedAsync(FeedType
feed = FeedType.hashPassport)
        {
            var status = await GetFeedsStatusAsync(feed)
                ?? throw new Exception(
                    $"Время обновления {feed} не получено.");
            return DateTime.Parse(status.UploadDatetime);
        }
        public static async Task LoadFeeds(string path)
        {
            Trace.WriteLine("Получение фидов...");
            Directory.CreateDirectory(path);
            foreach (var feed in Enum.GetValues<FeedType>())
            {
                var date = await GetUpdatedAsync(feed);
                Trace.WriteLine($"{date:g} {feed}");
                await DownloadFeedsAsync(feed, path);
            }
            string currentDate = DateTime.Now.ToString("yyyyMMdd");
            var connector = new
PGSQLConnector("Host=localhost;Database=users;Username=postgres;Pa
ssword=shkola77dan");

```

Класс Feeds.cs содержит все основные методы для работы с фидами (данными).

Метод GetFeedsStatusAsync() получает информацию о статусе указанного фида.

```

        public static async Task<FeedsStatus>
GetFeedsStatusAsync(FeedType feed)
        {
            string url = $"antifraud/feeds/{feed}";
            using var response = await TlsClient.GetAsync(url);
            response.EnsureSuccessStatusCode();
            return await
response.Content.ReadFromJsonAsync<FeedsStatus>()

```

```

        ?? throw new FeedsException("Статус фидов не получен.");
    }

```

Метод DownloadsFeedsAsync() загружает фиды (данные) в формате CSV и сохраняет их в указанную директорию.

```

    public static async Task DownloadFeedsAsync(FeedType
feed, string path)
    {
        Directory.CreateDirectory(path);
        string url = $"antifraud/feeds/{feed}/download";
        using var response = await TlsClient.GetAsync(url);
        response.EnsureSuccessStatusCode();
        string name = feed switch
        {
            FeedType.accountNumber => "account_number",
            FeedType.cardNumber     => "card_number",
            FeedType.ewalletNumber  => "ewallet_number",
            FeedType.fastPayNumber  => "fastpay_number",
            FeedType.hashPassport   => "passport_hash",
            FeedType.hashSnils      => "snils_hash",
            FeedType.inn            => "inn",
            FeedType.phoneNumber    => "phone_number",
            FeedType.swift          => "swift",
            _                       => feed.ToString()
        };
        string file = Path.Combine(path, name + ".csv");
        using var output = File.Create(file);
        await response.Content.CopyToAsync(output);
    }
}

```

Класс PGSQLConnector.cs предназначен для работы с PostgreSQL, предоставляющий функционал для загрузки данных в БД, экспорта данных в CSV, управление фидами МВД, ФинЦЕРТ и ИОС.

Метод LoadMvdTypeTable() загружает информацию о загруженных фидах МВД.

					ПГУ 1.090301.09.001 ПЗ	Лист
						55
Изм.	Лист	№ докум.	Подпись	Дата		

```

        public void LoadMvdTypeTable(string MvdHrid, string
currentDate)
    {
        using (var connection = new
NpgsqlConnection(_connectionString))
        {
            connection.Open();
            var insertMvdCommand = @"INSERT INTO
public.users_FeedsMvdDateDownloads (date, ""nameFeedsTrim"",
feedstype_id)
                                VALUES
('{currentDate}','{MvdHrid}', 2)";
            using (var cmd = new NpgsqlCommand(insertMvdCommand,
connection))
            {
                cmd.ExecuteNonQuery();
            }
        }
    }

```

Метод LoadFinCertDateDownloads() регистрирует загрузку фидов ФинЦерт.

```

        public void LoadFinCertDateDownloads(string currentDate)
    {
        using (var connection = new
NpgsqlConnection(_connectionString))
        {
            connection.Open();
            var insertFinCertDateCommand = @"INSERT INTO
public.users_Feedsfincertdatedownloads (date, feedstype_id)
VALUES ('{currentDate}', 1)";
            using (var cmd = new
NpgsqlCommand(insertFinCertDateCommand, connection))
            {
                cmd.ExecuteNonQuery();
            }
        }
    }

```

Метод LoadIOCDateDownloads() регистрирует обработку IOC-данных.

					ПГУ 1.090301.09.001 ПЗ	Лист
						56
Изм.	Лист	№ докум.	Подпись	Дата		

```

        public void LoadIOCDateDownloads(string IOCHrid, string
currentDate, string dateMailing)
    {
        using (var connection = new
NpgsqlConnection(_connectionString))
        {
            connection.Open();
            var insertIOCDateCommand = @"INSERT INTO
public.users_fincertiocdateprocessing ("dateMailing",
"ioctrim", feedstype_id, date)
VALUES ('{dateMailing}', '{IOCHrid}', 3, '{currentDate}');";
            using (var cmd = new NpgsqlCommand(insertIOCDateCommand,
connection))
            {
                cmd.ExecuteNonQuery();
            }
        }
    }

```

Метод ImportToDatebaseFromCsvFilesBach() формирует пакетную команду для импорта.

```

        private string
ImportToDatebaseFromCsvFilesBach(NpgsqlConnection connection,
string tempTableName, TableImportConfig config)
    {
        var columns = new List<string>();
        foreach (var mapping in config.ColumnMappings)
        {
            // Определяем тип для каждой колонки
            var columnType = mapping.Value == "count" ? "INTEGER" :
"TEXT";
            columns.Add($"\"{mapping.Value}\" {columnType}");
        }
        var createTempTable = $"CREATE TEMP TABLE {tempTableName}
({string.Join(", ", columns)});";
        var csvColumns =
config.ColumnMappings.Keys;//.Concat(config.KeyColumns ??
Enumerable.Empty<string>());
    }

```

```

var dbColumns =
config.ColumnMappings.Values;//.Concat(config.KeyColumns ??
Enumerable.Empty<string>());

var importToTempTable = @"COPY
{tempTableName} (""{string.Join "\", \"", dbColumns)}"")
        FROM '{config.CsvFilePath}'
        DELIMITER ','
        CSV HEADER
        ENCODING 'UTF-8';";

if (config.KeyColumns == null || !config.KeyColumns.Any())
    throw new ArgumentException("Не указаны ключевые колонки
для проверки дубликатов");

var joinConditions = config.KeyColumns
    .Select(k => $"t.\"{k}\" = s.\"{k}\"");
var insertColumns = config.ColumnMappings.Values
    .Select(c => $"\"{c}\"");
var selectValues = config.ColumnMappings.Values
    .Select(c => $"s.\"{c}\"");
var insertNewRecords = @"
        INSERT INTO public.\"{config.TableName}\" ({string.Join(",
\", insertColumns)})

        SELECT {string.Join(", ", selectValues)}
        FROM {tempTableName} s
        WHERE NOT EXISTS (
            SELECT 1
            FROM public.\"{config.TableName}\" t
            WHERE {string.Join(" AND ", joinConditions)});";
var updateDateExcludes = @"UPDATE public.\"{config.TableName}\" t
    SET "dateExcluded" = CURRENT_DATE
    WHERE NOT EXISTS (
        SELECT 1
        FROM {tempTableName} s
        WHERE {string.Join(" AND ", joinConditions)})
    AND t."dateExcluded" is NULL;";

```

```

        var bachCommand = createTempTable + "\n" + importToTempTable +
"\n" + insertNewRecords + "\n" + updateDateExcludes;
        return bachCommand;
    }

```

Класс SFTPConnector.cs предоставляет удобную обертку для работы с SFTP-сервером, реализуя основные операции по передаче файлов с использованием библиотеки Renci.SshNet.

```

using System;
using System.IO;
using Renci.SshNet;
using Renci.SshNet.Sftp;
namespace FincertClient;
public class SftpFileTransfer : IDisposable
{
    private readonly SftpClient _sftpClient;
    private bool _disposed = false;
    public SftpFileTransfer(string host, int port, string
username, string privateKeyPath)
    {
        var keyFiles = new[] { new PrivateKeyFile(privateKeyPath)
};
        _sftpClient = new SftpClient(host, port, username,
keyFiles);
    }
    public void Connect()
    {
        if (_sftpClient.IsConnected)
            return;
        _sftpClient.Connect();
    }
    public void Disconnect()
    {
        if (_sftpClient.IsConnected)
            _sftpClient.Disconnect();
    }
}

```

```

    public bool IsConnected => _sftpClient.IsConnected;
    public void UploadFile(string localFilePath, string
remoteFilePath, bool overwrite = true)
    {
        ValidateConnection();
        if (!File.Exists(localFilePath))
            throw new FileNotFoundException($"Локальный файл не
найден: {localFilePath}");
        if (_sftpClient.Exists(remoteFilePath) && !overwrite)
            throw new IOException($"Удаленный файл уже существует:
{remoteFilePath}");
        using var fileStream = File.OpenRead(localFilePath);
        _sftpClient.UploadFile(fileStream, remoteFilePath);
    }
    public void DownloadFile(string remoteFilePath, string
localFilePath, bool overwrite = true)
    {
        ValidateConnection();

        if (!_sftpClient.Exists(remoteFilePath))
            throw new FileNotFoundException($"Удаленный файл не
найден: {remoteFilePath}");
        if (File.Exists(localFilePath) && !overwrite)
            throw new IOException($"Локальный файл уже существует:
{localFilePath}");
        using var fileStream = File.Create(localFilePath);
        _sftpClient.DownloadFile(remoteFilePath, fileStream);
    }
    public void DeleteFile(string remoteFilePath)
    {
        ValidateConnection();
        if (!_sftpClient.Exists(remoteFilePath))
            throw new FileNotFoundException($"Удаленный файл не
найден: {remoteFilePath}");
        _sftpClient.DeleteFile(remoteFilePath);
    }

```

```

    }

    public IEnumerable<SftpFile> ListDirectory(string
remoteDirectoryPath)
    {
        ValidateConnection();
        return
(IEnumerable<SftpFile>)_sftpClient.ListDirectory(remoteDirectoryPa
th);
    }
    private void ValidateConnection()
    {
        if (!_sftpClient.IsConnected)
            throw new InvalidOperationException("Не подключено к
SFTP серверу. Вызовите метод Connect() перед выполнением
операций.");
    }
    public void Dispose()
    {
        Dispose(true);
        GC.SuppressFinalize(this);
    }
    protected virtual void Dispose(bool disposing)
    {
        if (_disposed)
            return;
        if (disposing)
        {
            Disconnect();
            _sftpClient.Dispose();
        }
        _disposed = true;
    }
    ~SftpFileTransfer()
    {
        Dispose(false);
    }

```

```

    }
}

```

Остальной код и методы приведены в приложении Б (листинг программы).

1.4.3 Описание веб-приложения

Для создания веб-приложения использовался фреймворк Django для веб-разработки.

Файл settings.py отвечает за настройки веб-приложения, настройка для работы с базами данных и статическими файлами и подключения приложений.

```

from pathlib import Path
BASE_DIR = Path(__file__).resolve().parent.parent
DEBUG = True
ALLOWED_HOSTS = ['fincertwebapi.ru', 'localhost',
'127.0.0.1']
INSTALLED_APPS = [
'django.contrib.admin',
'django.contrib.auth',
'django.contrib.contenttypes',
'django.contrib.sessions',
'django.contrib.messages',
'django.contrib.staticfiles',
'users'
]
MIDDLEWARE = [
'django.middleware.security.SecurityMiddleware',
'django.contrib.sessions.middleware.SessionMiddleware',
'django.middleware.common.CommonMiddleware',
'django.middleware.csrf.CsrfViewMiddleware',
'django.contrib.auth.middleware.AuthenticationMiddleware',
'django.contrib.messages.middleware.MessageMiddleware',
'django.middleware.clickjacking.XFrameOptionsMiddleware',
]
ROOT_URLCONF = 'prdip.urls'
TEMPLATES = [

```

```

        {
            'BACKEND':
'django.template.backends.django.DjangoTemplates',
            'DIRS': [],
            'APP_DIRS': True,
            'OPTIONS': {
                'context_processors': [
                    'django.template.context_processors.request',
                    'django.contrib.auth.context_processors.auth',

'django.contrib.messages.context_processors.messages',

                ],
WSGI_APPLICATION = 'prdip.wsgi.application'
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME': 'users',
        'USER': 'postgres',
        'PASSWORD': 'shkola77dan',
        'HOST': 'localhost',
        'PORT': '5432',
    }
}
AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilar
ityValidator'
    },
    {
        'NAME':
'django.contrib.auth.password_validation.MinimumLengthValidat
o
        'NAME':
'django.contrib.auth.password_validation.CommonPasswordValida

```

```
tor',
'NAME':
'django.contrib.auth.password_validation.NumericPasswordValidator'
},
]
```

```
LANGUAGE_CODE = 'ru'
TIME_ZONE = 'UTC'
USE_I18N = True
USE_TZ = True
STATIC_URL = '/static/'
STATIC_ROOT = "D:/nginx/static/"
DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
```

Файл `url.py` отвечает за определение корневых маршрутов всего проекта, связывает URL-адреса с соответствующими view-функциями, организует маршрут между разными приложениями проекта.

```
from django.contrib import admin
from django.urls import path, include
urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('users.urls')),
]
```

Файл `views.py` определяет, какие страницы будут присутствовать на вебе-сайте, а также предоставляет логику для передачи информации на каждую страницу.

```
from django.core.paginator import Paginator
from django.shortcuts import render
from .models import *
from django.db.models import Q
def unified_feeds_view(request):
    items_per_page = 50
    active_tab = request.GET.get('tab', 'fincert')
    date_filter = request.GET.get('date_filter', '')
    fincert_dates =
FeedsFinCertDateDownloads.objects.select_related('feedstype').orde
```

```

r_by('-date')[:10]

def get_filtered_queryset(model, date_field='dateFixed'):
    queryset =
    model.objects.select_related('datedownloads').all()
    if date_filter:
        queryset =
        queryset.filter(**{f'{date_field}__icontains': date_filter})
    return queryset.order_by(f'-{date_field}')

account_numbers = get_filtered_queryset(FeedsAccountNumbers)
card_numbers = get_filtered_queryset(FeedsCardNumbers)
ewallet_numbers = get_filtered_queryset(FeedsEwalletNumbers)
fastpay_numbers = get_filtered_queryset(FeedsFastpayNumbers)
inn_numbers = get_filtered_queryset(FeedsInn)
passport_hashes = get_filtered_queryset(FeedsPassportHash)
phone_numbers = get_filtered_queryset(FeedsPhoneNumbers)
snils_hashes = get_filtered_queryset(FeedsSnilsHash)
swift_data = get_filtered_queryset(FeedsSwift)
mvd_dates = FeedsMvdDateDownloads.objects.all().order_by('-
date')[:10]

mvd_accounts = get_filtered_queryset(FeedsMvdAccountNumbers,
'datedownloads__date')

mvd_cards = get_filtered_queryset(FeedsMvdCardNumbers,
'datedownloads__date')

mvd_fastpay = get_filtered_queryset(FeedsMvdFastPayNumbers,
'datedownloads__date')

mvd_inn = get_filtered_queryset(FeedsMvdInn,
'datedownloads__date')

mvd_passports = get_filtered_queryset(FeedsMvdPassportHash,
'datedownloads__date')

ioc_data =
FinCertIocDateProcessing.objects.select_related('feedstype').order
_by('-dateMailing')[:10]

def paginate_data(request, data, param_name):
    paginator = Paginator(data, items_per_page)
    page_number = request.GET.get(f'{param_name}_page')

```

```

        return paginator.get_page(page_number)
    context = {
        'active_tab': active_tab,
        'date_filter': date_filter,
        'fincert_dates': fincert_dates,
        'account_numbers_page': paginate_data(request,
account_numbers, 'account'),
        'card_numbers_page': paginate_data(request, card_numbers,
'card'),
        'ewallet_numbers_page': paginate_data(request,
ewallet_numbers, 'ewallet'),
        'fastpay_numbers_page': paginate_data(request,
fastpay_numbers, 'fastpay'),
        'inn_numbers_page': paginate_data(request, inn_numbers,
'inn'),
        'passport_hashes_page': paginate_data(request,
passport_hashes, 'passport'),
        'phone_numbers_page': paginate_data(request,
phone_numbers, 'phone'),
        'snils_hashes_page': paginate_data(request, snils_hashes,
'snils'),
        'swift_data_page': paginate_data(request, swift_data,
'swift'),
        'mvd_dates': mvd_dates,
        'mvd_accounts': mvd_accounts,
        'mvd_cards': mvd_cards,
        'mvd_fastpay': mvd_fastpay,
        'mvd_inn': mvd_inn,
        'mvd_passports': mvd_passports,
        'ioc_data': ioc_data,
    }
    return render(request, 'users/unified_view.html', context)

```

Файл unified_view.html отвечает за создания каркаса нашего приложения с помощью HTML кода, код можно посмотреть в приложении Б (листинг программы).

1.4.4 Руководство пользователя

Чтобы начать работу с программой, необходимо настроить все файлы конфигурации, ввести в них логины и пароли от личного кабинет ФинЦерт, ввести отпечаток сертификата пользователя для получения доступа к личному кабинету, после настройки конфигурации, жмем на кнопку «Сохранить». Окно для настройки конфигурации представлено на рисунках - 10, 11, 12.

					ПГУ 1.090301.09.001 ПЗ	Лист
						67
Изм.	Лист	№ докум.	Подпись	Дата		

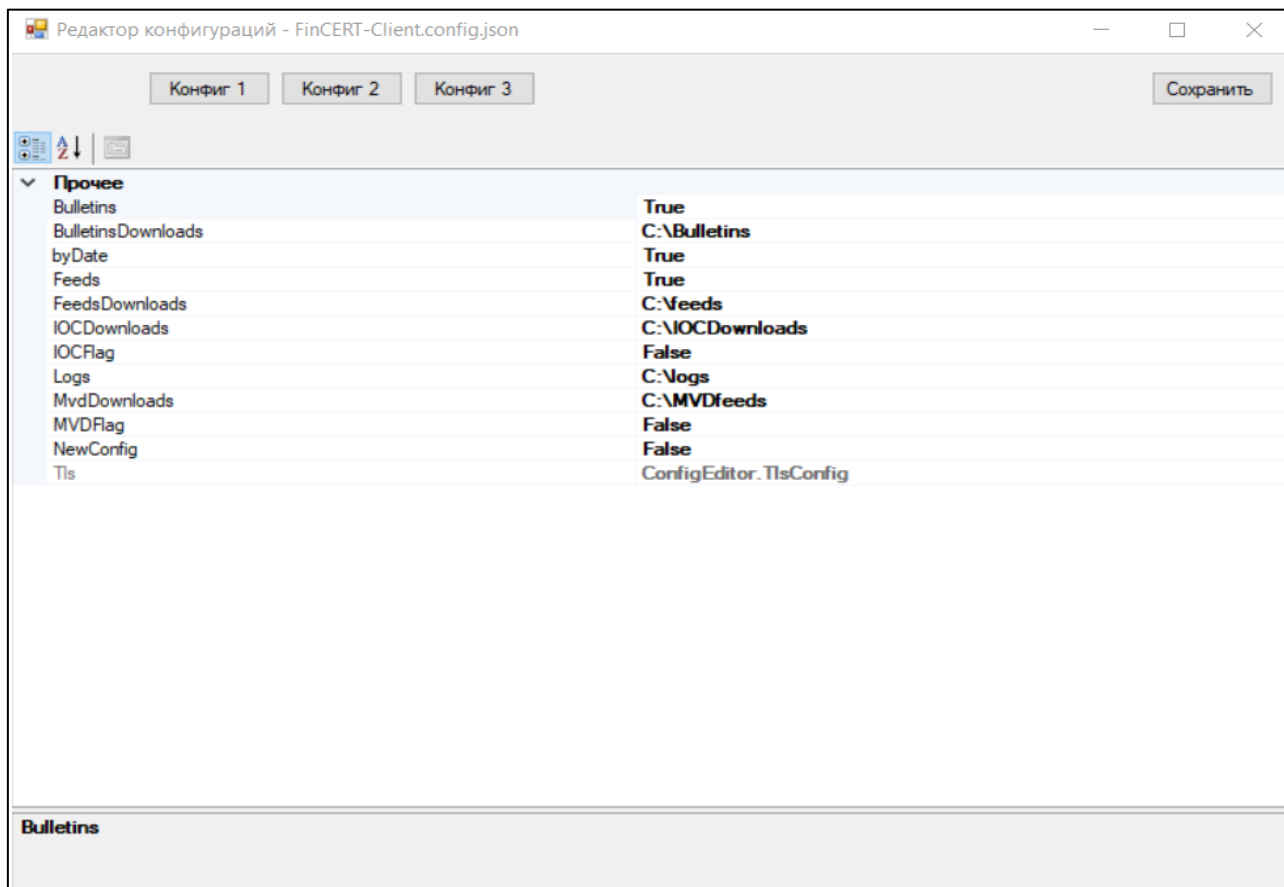


Рисунок 10 — Окно для редактирования конфигурации 1

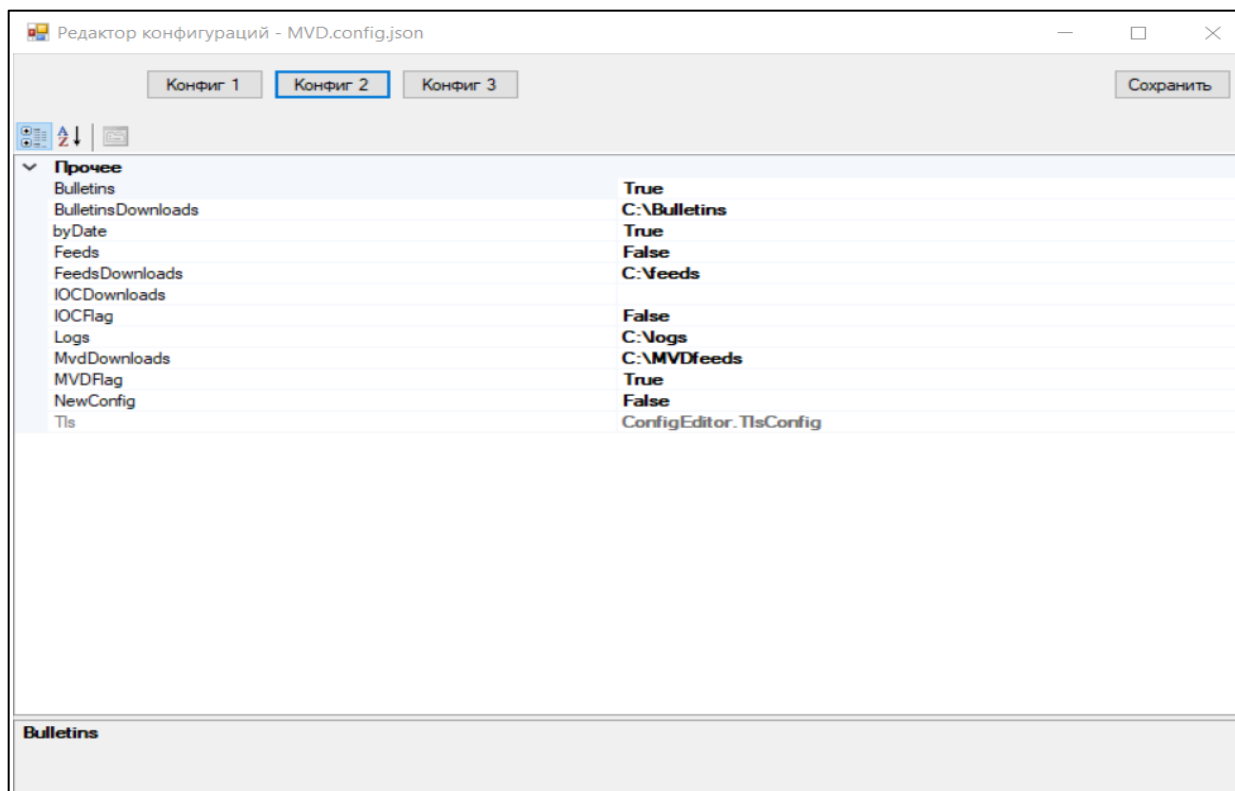


Рисунок 11 — Окно для редактирования конфигурации 2

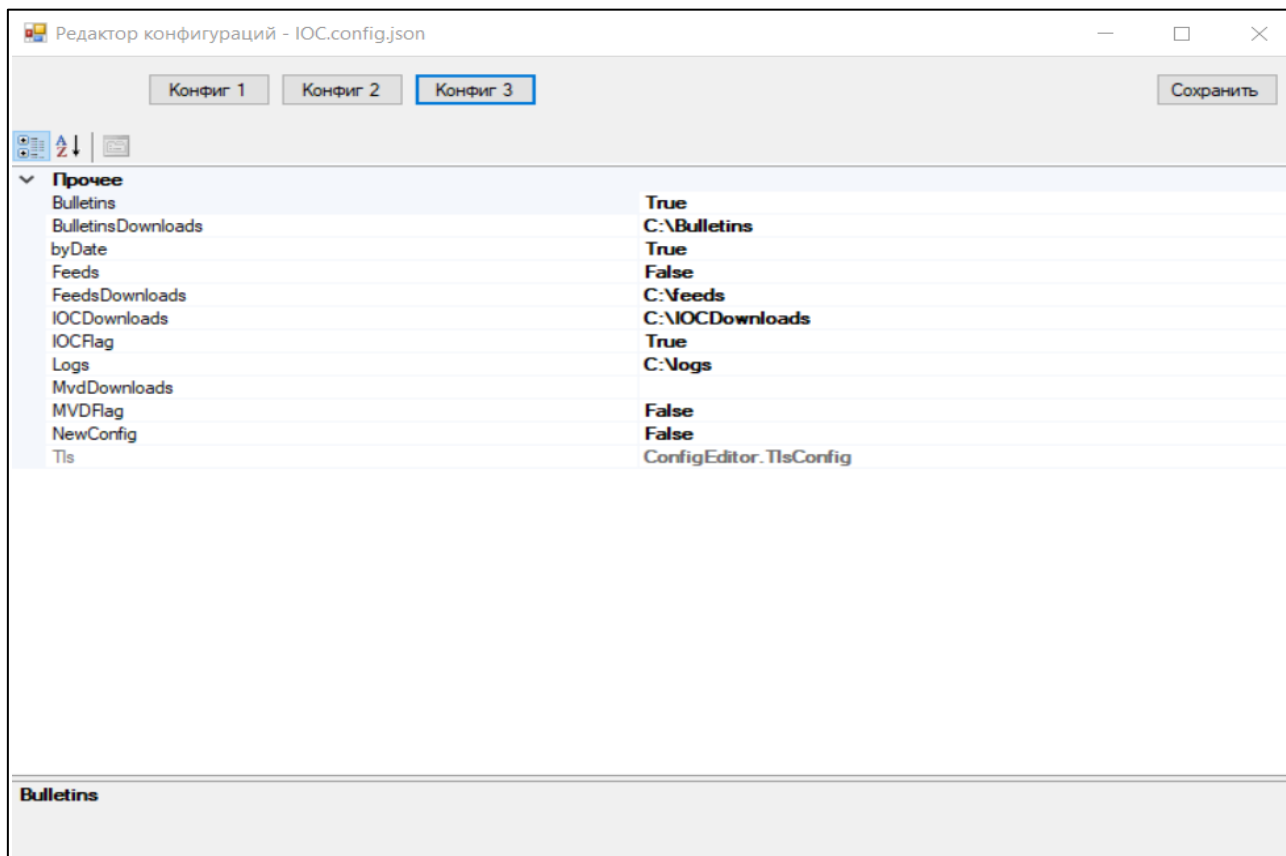


Рисунок 12 — Окно для редактирования конфигурации 3

Для запуска программы используется «Планировщик заданий» Windows, чтобы настроить автоматический запуск программы в определенное время. Окно планировщика задач представлено на рисунке 13.

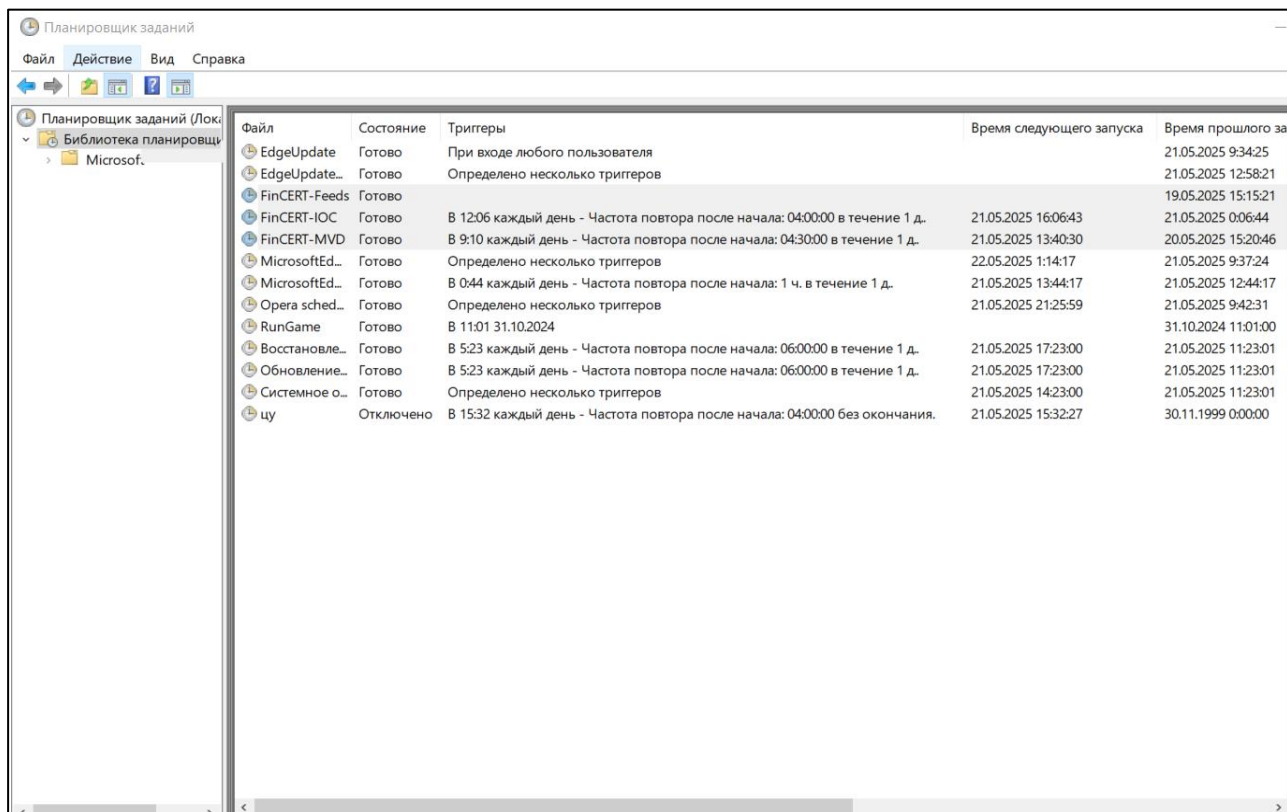


Рисунок 13 — Окно «Планировщик заданий» Windows

После запуска программы откроется окно консоли PowerShell (рисунок 14), в котором будет выводиться статус работы программы и код возврата, если код возврата 0, то программа отработала правильно и без ошибок.



Рисунок 14 — Окно вывода при запуске программы

После работы программы можно отследить, как она отработала в файла формата .log. Файл с логами представлен на рисунке 15.

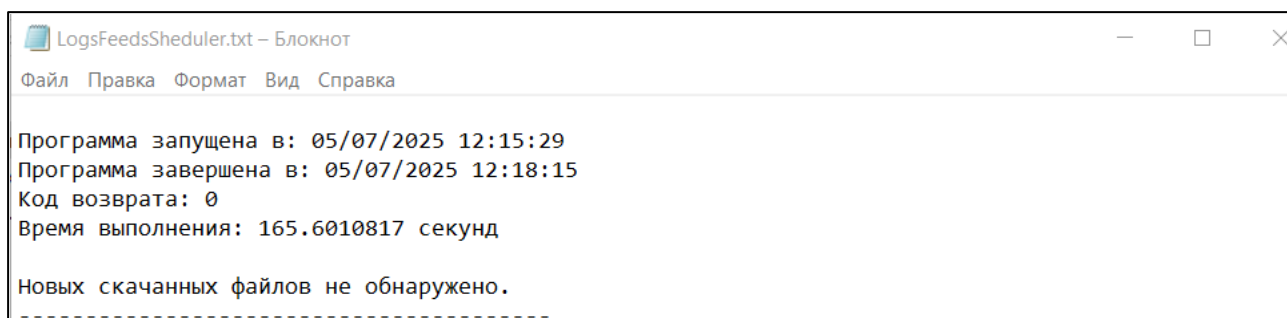


Рисунок 15 — Файл фиксирующий работу программы

В случае неудачной работы программы в файле будет указан код возврата, отличный от 0. Файл с лога с неудачной работой программы представлено на рисунке 16.

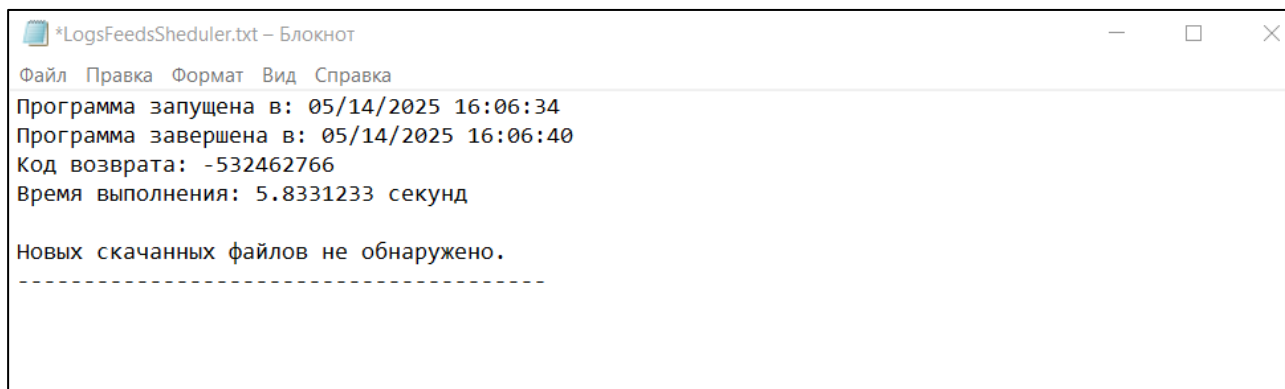


Рисунок 16 — Результат запись в файл .log после ошибки работы программы

Когда срабатывает триггер, службы с нашей программой открывается консоль, в которой происходит вся работа программы. Начало работы программы можно посмотреть на рисунке 17.

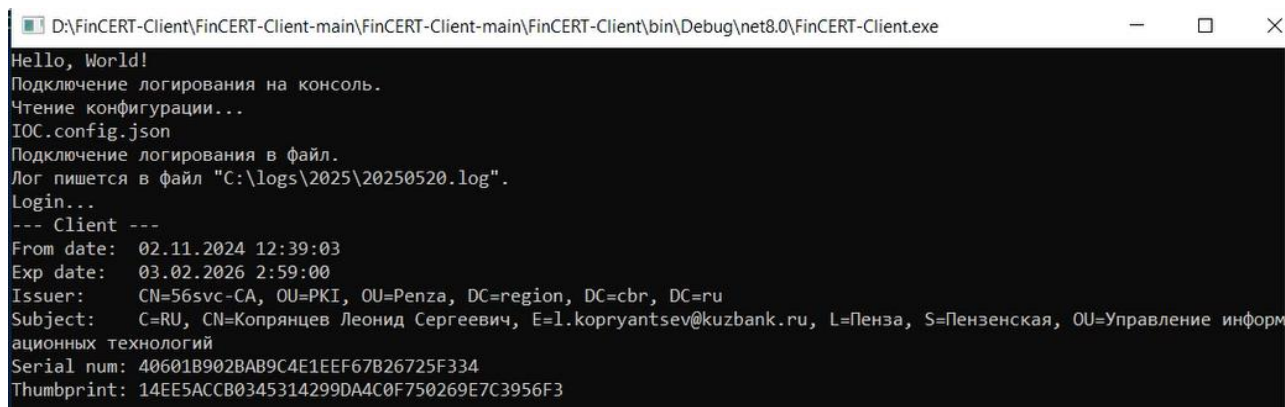


Рисунок 17 — Начало работы программы

Процесс начала скачивания файлов на рисунке 18, выводится в консоль для понимания, какие файлы начали скачивание.



Рисунок 18 — Процесс скачивания файлов с АСОИ ФинЦЕРТ

После скачивания файлов, они обрабатываются и загружаются в БД. Начало загрузки файлов в БД представлено на рисунке 19.

```

Файл: ewallet_number.csv
-----Импорт в БД успешен-----
-----Начало импорта в БД-----

Обработка таблицы: users_feedsfastpaynumbers
Файл: fastpay_number.csv
-----Импорт в БД успешен-----
-----Начало импорта в БД-----

Обработка таблицы: users_feedsinn
Файл: inn.csv
-----Импорт в БД успешен-----
-----Начало импорта в БД-----

Обработка таблицы: users_feedspassporthash
Файл: passport_hash.csv
-----Импорт в БД успешен-----
-----Начало импорта в БД-----

Обработка таблицы: users_feedsphonenumbers
Файл: phone_number.csv
-----Импорт в БД успешен-----
-----Начало импорта в БД-----

Обработка таблицы: users_feedssnilshash
Файл: snils_hash.csv
-----Импорт в БД успешен-----

```

Рисунок 19 — Процесс загрузки файлов в БД

Для файлов МВД, результат работы программы отличается от файлов МВД. Файлы МВД, проверяются на текущую дату, если дата не совпадает с запрошенной, то файл будет пропущен, это можно увидеть на рисунке 20.

```

Получение бюллетеней по папкам...
Всего: 3568, в листе: 20.
FinCERT-20250520-01-FEEDS
Получен фид МВД 'feeds_20250520-01.zip'
Пропущен бюллетень FinCERT-20250519-06 - не соответствует текущей дате
Пропущен бюллетень FinCERT-20250519-04 - не соответствует текущей дате
Пропущен бюллетень FinCERT-20250518-04 - не соответствует текущей дате
Пропущен бюллетень FinCERT-20250517-03 - не соответствует текущей дате
Пропущен бюллетень FinCERT-20250516-01-Дайджест - не соответствует текущей дате

```

Рисунок 20 — Результат работа программы с файлами от МВД

После скачивания всех файлов от МВД, начинается загрузка файлов в БД. Результат загрузки файлов от МВД в БД представлен на рисунке 21.

```

-----Начало импорта в БД-----
Обработка таблицы: users_feedsmvdaccountnumber
Файл: account_number.csv
-----Импорт в БД успешен-----
-----Начало импорта в БД-----
Обработка таблицы: users_feedsmvdcardnumbers
Файл: card_number.csv
-----Импорт в БД успешен-----
-----Начало импорта в БД-----
Обработка таблицы: users_feedsmvdfastpaynumber
Файл: fastpay_number.csv
-----Импорт в БД успешен-----
-----Начало импорта в БД-----
Обработка таблицы: users_feedsmvdinn
Файл: inn.csv
-----Импорт в БД успешен-----
-----Начало импорта в БД-----
Обработка таблицы: users_feedsmvdpassporthash
Файл: passport_hash.csv
-----Импорт в БД успешен-----

```

Рисунок 21 — Результат загрузки файлов от МВД в БД

Для службы, отвечающей за скачивание файлов ИОС, после скачивания файла, начинает его обработку. Результат работы службы с файлами ИОС можно посмотреть на рисунке 22.

```

Получение бюллетеней по папкам...
Всего: 3568, в листе: 20.
FinCERT-20250520-04
Получен ИОС файл 'FinCERT_20250520-143555.zip'
20250520
C:\IOCD\Downloads\FinCERT_20250520-143555.zip
Обработано 157 записей
2025-05-20 17:28:50 [Info]: Total blocked IPs: 22303
2025-05-20 17:29:08 [Info]: Total blocked CIDR IPs: 10331092
2025-05-20 17:29:08 [Warning]: Unknown IOC type: 'Type_IOC'
2025-05-20 17:29:08 [Warning]: Error resolving hostname 'atomicsmet.run': Этот хост неизвестен.
2025-05-20 17:29:09 [Warning]: Error resolving hostname 'datawavej.digital': Этот хост неизвестен.
2025-05-20 17:29:18 [Warning]: Error resolving hostname 'medikosmos.ru': Обычно - это временная ошибка, возникающая
время разрешения имени узла, и означающая, что локальный сервер не получил ответа от полномочного сервера.
2025-05-20 17:29:28 [Warning]: Error resolving hostname 'modnobutik.ru': Обычно - это временная ошибка, возникающая
время разрешения имени узла, и означающая, что локальный сервер не получил ответа от полномочного сервера.
2025-05-20 17:29:39 [Warning]: Error resolving hostname 'ns2.taipower.energy': Этот хост неизвестен.
2025-05-20 17:29:39 [Warning]: Error resolving hostname 'racxilb.digital': Этот хост неизвестен.
2025-05-20 17:29:39 [Warning]: Error resolving hostname 'techchaiun.live': Этот хост неизвестен.
2025-05-20 17:29:39 [Warning]: Error resolving hostname 'testcawep.run': Этот хост неизвестен.
2025-05-20 17:29:49 [Warning]: Error resolving hostname 'zelenotex.ru': Обычно - это временная ошибка, возникающая в
время разрешения имени узла, и означающая, что локальный сервер не получил ответа от полномочного сервера.
2025-05-20 17:29:49 [Info]: Total IPs to check: 160

```

Рисунок 22 — Результат работы службы с файлами ИОС

После выполнения обработки файлов ИОС должен получиться архив с текстовым файлом, в котором содержатся все обработанные IP-адреса, все

					ПГУ 1.090301.09.001 ПЗ	Лист
						73
Изм.	Лист	№ докум.	Подпись	Дата		

обработанные подсети, и доменные имена. Результат обработки файлов ИОС можно посмотреть на рисунке 23.

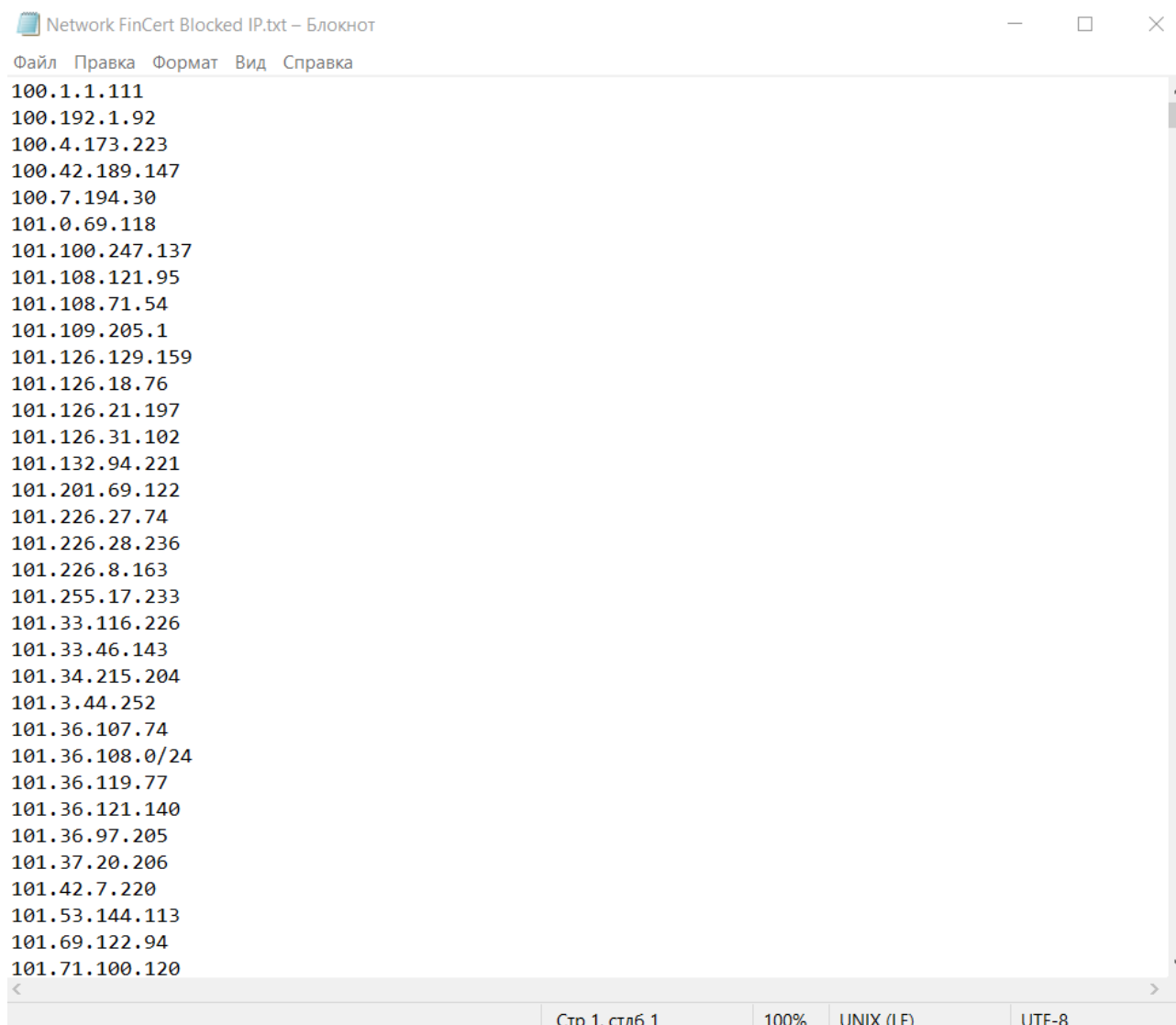


Рисунок 23 — Результат обработки файлов ИОС

После того, как файлы ИОС были обработаны, программы загружает их, на SFTP-сервер. Результат подключения можно посмотреть на рисунке 24 и результат загрузки на SFTP-сервер на рисунке 25.

```
Подключение к SFTP серверу успешно!
Файл успешно загружен
Файл успешно загружен
```

Рисунок 24 — Результат успешного подключения и загрузки на SFTP-сервер

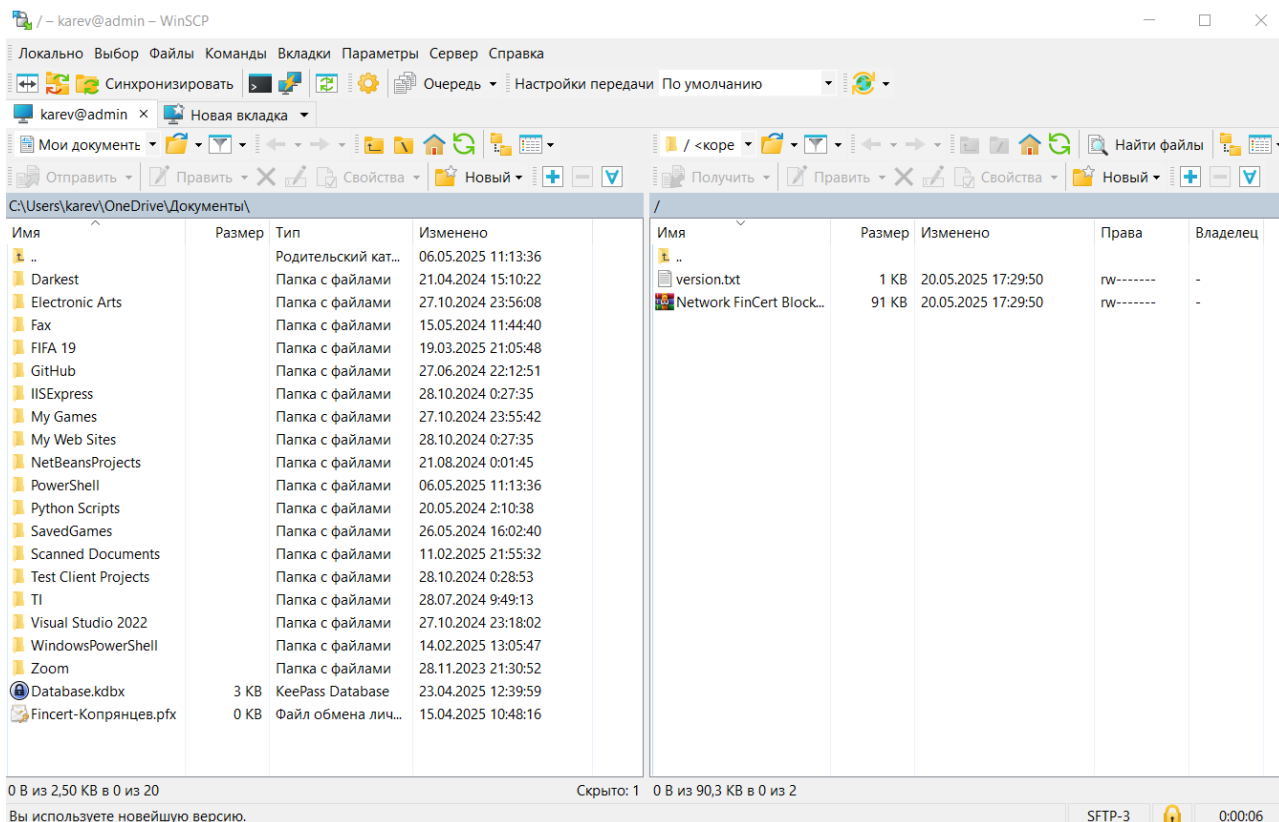


Рисунок 25 — Результат загрузки на SFTP-сервер

После того, как все файлы были обработаны и загружены в БД, их можно просмотреть в веб-приложении, если перейти по адресу fincertwebapi.ru. Главная страница представлена на рисунке 26.

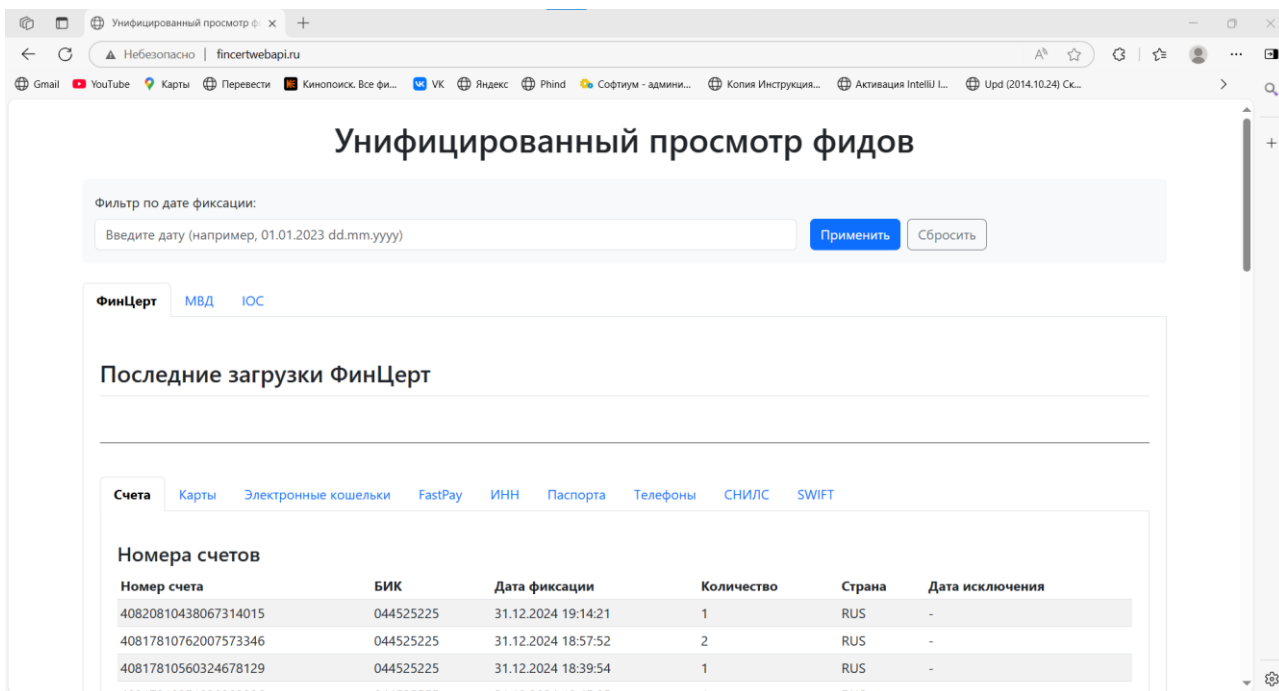


Рисунок 26 — Главная страница веб-приложения

В данном веб-приложении можно использовать строку поиска, она представлена на рисунке 27, для нахождения определенных данных, на указанную дату.

Фильтр по дате фиксации:

31.10.2024

Применить

Сбросить

ФинЦерт

МВД

ИОС

Последние загрузки ФинЦерт

Счета

Карты

Электронные кошельки

FastPay

ИНН

Паспорта

Телефоны

СНИЛС

SWIFT

Номера счетов

Номер счета	БИК	Дата фиксации	Количество	Страна	Дата исключения
40903810600002006348	044525068	31.10.2024 22:31:18	1	RUS	-
40817810190420015373	044030790	31.10.2024 22:28:04	1	RUS	-
40817810550005078131	044525225	31.10.2024 22:20:19	1	RUS	-

Рисунок 27 — Результат работы строки поиска

Также присутствует возможность просмотра всех файлов с удобным интерфейсом для пользования на рисунке 28.

40817810051037135607	044525555	31.10.2024 10:43:22	1	RUS	-
40817810710480065858	044525593	31.10.2024 10:37:41	2	RUS	-
40817810151036952448	044525555	31.10.2024 10:28:13	1	RUS	-
40817810751033030747	044525555	31.10.2024 10:09:15	1	RUS	-
40817810351037022290	044525555	31.10.2024 10:02:44	1	RUS	-
40817810506750115985	044525593	31.10.2024 10:02:22	1	RUS	-
40817810205613352898	044525593	31.10.2024 08:42:00	1	RUS	-
40817810150186827881	043469743	31.10.2024 04:27:14	2	RUS	-
40817810510096045734	043601968	31.10.2024 04:19:17	2	RUS	-
40817810963800009829	044525111	31.10.2024 04:15:12	1	RUS	-
40817810890039024148	040813713	31.10.2024 00:32:43	1	RUS	-

«

1

2

»

Рисунок 28 — Удобный интерфейс для перехода между страницами

Таблицу с датой последней обработки ИОС файлов можно посмотреть на рисунке 29.

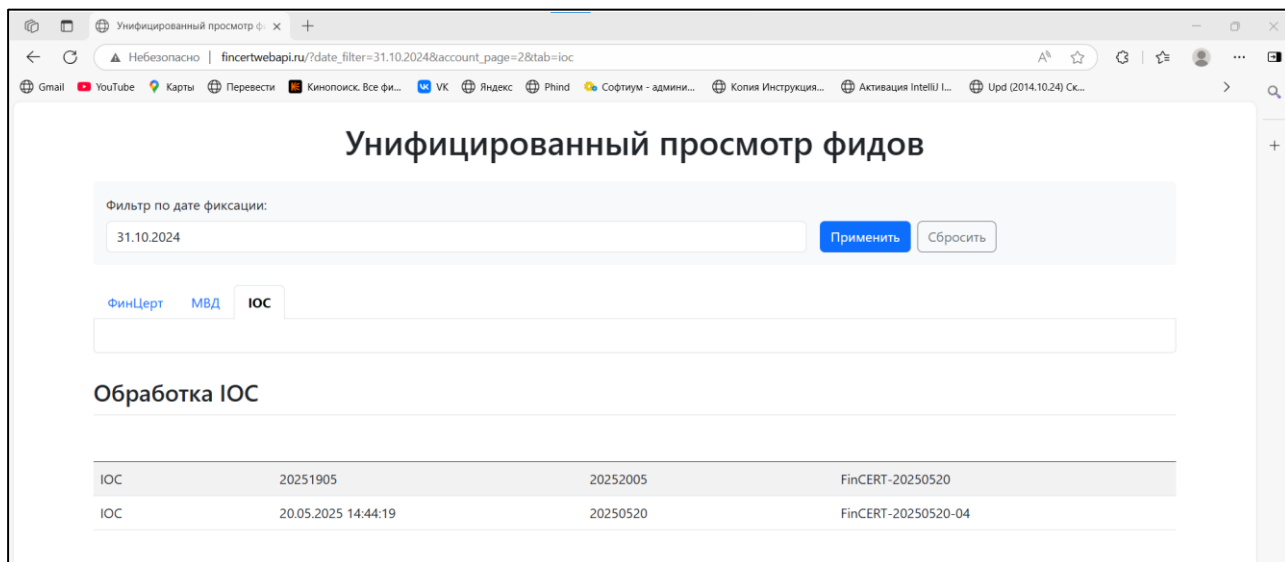


Рисунок 29— Таблица с датой обработки файлов ИОС

Для администрирования веб-приложения нужно перейти по адресу fincertwebapi.ru/admin для перехода на страницу авторизации администратора. Окно авторизации администраторов представлена на рисунке 30.

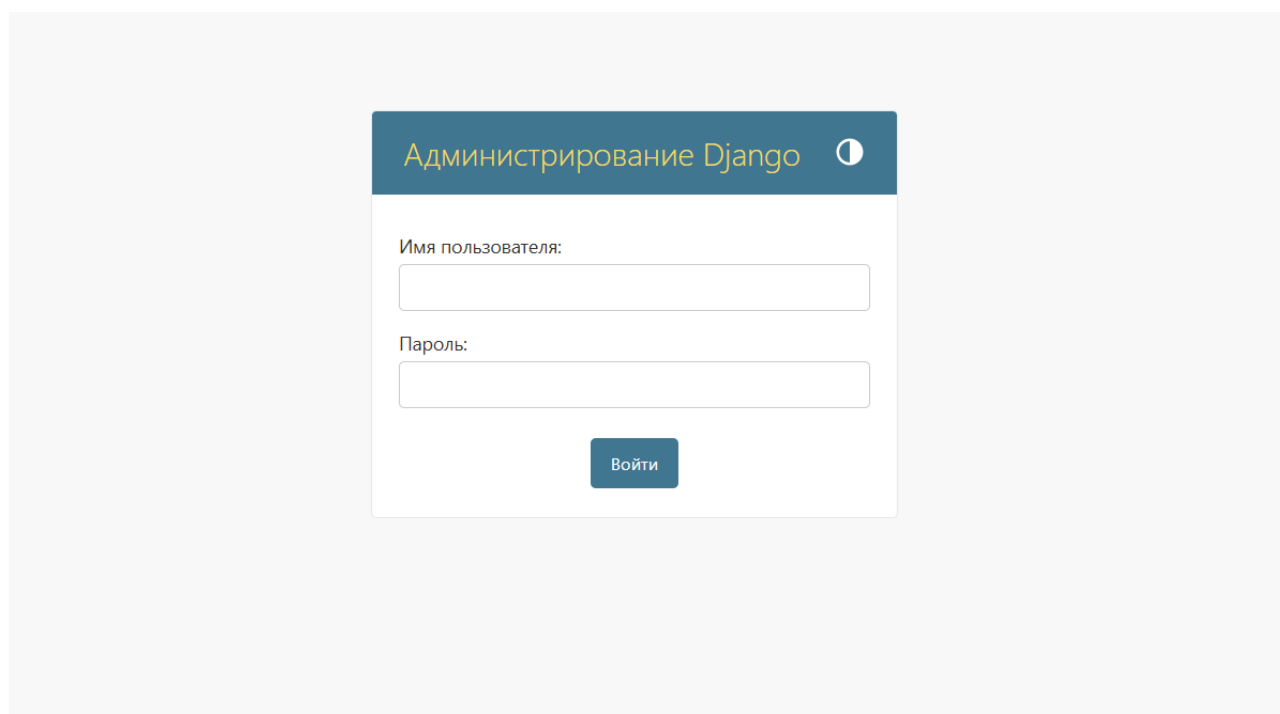


Рисунок 30 — Окно авторизации для администраторов

После успешной авторизации можно вручную редактировать таблицы веб-приложения, управлять пользователями, давать им различные статусы (рисунок 31).

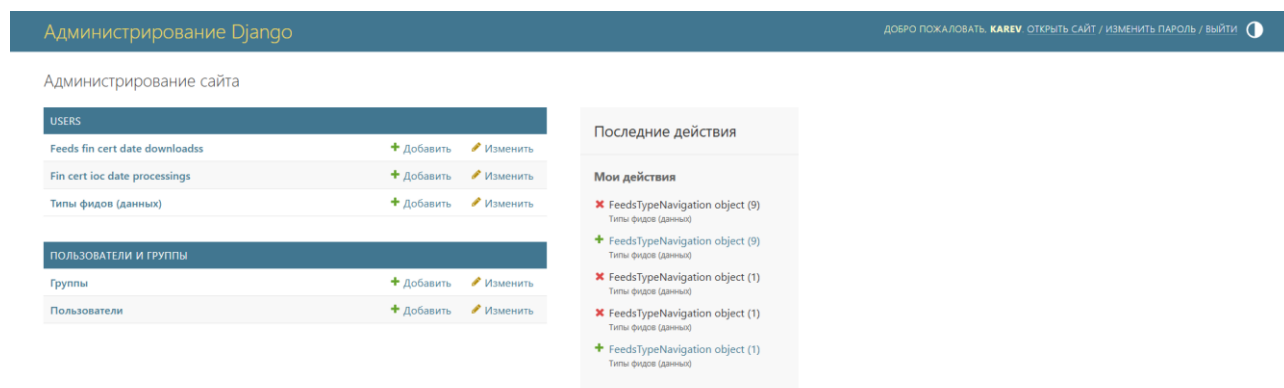


Рисунок 31 — Панель администрирования

2 Охрана труда

Современное законодательство в области охраны труда при работе с электронно-вычислительными машинами (ЭВМ) выдвигает следующие требования [12]:

Помещение, в котором осуществляется работа с ПК, должно соответствовать следующим требованиям:

- освещение должно соответствовать стандартам, а в случае отсутствия естественного освещения должны быть предоставлены расчеты, доказывающие безопасность работы без него;

- оконные проемы должны иметь регулируемые устройства, такие как жалюзи или занавеси;

- компь;теры не должны размещаться в местах, где есть риск попадания влаги.

- рабочие места с компьютерами должны быть оборудованы защитным заземлением;

- в помещениях с компьютерами следует проводить ежедневную влажную уборку и проветривание каждый час;

- рекомендуемая температура воздуха в помещении с компьютерами составляет $+19^{\circ}\text{C}$ - $+21^{\circ}\text{C}$;

- рабочий стол должен быть спроектирован таким образом, чтобы обеспечить оптимальное расположение используемого оборудования в соответствии с выполняемой работой, разрешается использование различных типов рабочих столов, соответствующих современным стандартам эргономики;

- в помещениях с компьютерами следует проводить ежедневную влажную уборку и проветривание каждый час.

Рабочие места с персональными компьютерами должны быть организованы следующим образом:

- при организации рабочих мест с компьютерами необходимо учитывать, что расстояние между рабочими столами, на которых

установлены видеомониторы, должно быть не менее 2,0 метров в направлении между ними и не менее 1,2 метра между боковыми поверхностями видеомониторов;

- размещение компьютеров должно предусматривать расстояние не менее 1 метра от источников тепла. Следует избегать прямого попадания солнечного света на них;

- запрещается размещать рабочие места с компьютерами рядом с силовыми кабелями, высоковольтными трансформаторами и другим оборудованием, способным создавать помехи в работе компьютеров;

- при установке компьютеров следует избегать прохождения интерфейсных и электропитания кабелей по полу. В случае необходимости, их следует закрыть защитными коробками;

- экран видеомонитора должен находиться от глаз пользователя на расстоянии 600-700 мм, но не ближе 500 мм с учетом размеров алфавитно-цифровых знаков и символов;

- системный блок и монитор следует разместить так, чтобы расстояние от вентиляционных отверстий этих устройств до стены (или стенки мебели) составляло не менее 10 сантиметров;

- искусственное освещение в помещениях для эксплуатации компьютеров должно осуществляться системой общего равномерного освещения. Освещенность на поверхности рабочего стола в зоне размещения ПК должна быть 300-500 люкс. Освещение не должно создавать бликов на поверхности экрана. Освещенность поверхности экрана не должна быть более 300 люкс.

В случае работы на компьютере в течение 8-часовой смены следует предусматривать перерывы в соответствии с категорией работ:

- для работников I категории перерывы устанавливаются через 2 часа после начала смены и через 2 часа после обеденного перерыва продолжительностью 15 минут каждый;

- работники II категории должны иметь перерыв через 2 часа после начала смены и через 1,5 - 2,0 часа после обеденного перерыва продолжительностью 15 минут каждый или продолжительностью 10 минут через каждый час работы;

- для работников III категории рекомендуется устанавливать перерыв через 1,5 - 2,0 часа после начала смены и также через 1,5 - 2,0 часа после обеденного перерыва продолжительностью 20 минут каждый или продолжительностью 15 минут через каждый час работы.

При 12-часовой смене регламентированные перерывы должны быть предусмотрены в первые 8 часов работы так же, как и при 8-часовой смене. В течение последних 4 часов работы, независимо от категории и вида работ, необходимо предусматривать перерыв каждый час продолжительностью 15 минут (таблица 19).

Таблица 19 – Время регламентированных перерывов в зависимости от категории работы

Категория работы	Уровень нагрузки за рабочую смену при видах работ с компьютером			Суммарное время регламентированных перерывов, мин.	
	группа А, количество знаков	группа Б, количество знаков	группа В, час.	при 8-часовой смене	при 12-часовой смене
I	До 20000	До 15000	До 2,0	30	70
II	До 40000	До 30000	До 4,0	50	90
III	До 60000	До 40000	До 6,0	70	120

Заключение

В ходе реализации данной работы было изучено API центра взаимодействия и реагирования Департамента информационной безопасности «ФинЦЕРТ», методы преобразования и хранения табличных данных с помощью СУБД PostgreSQL, методы работы с SFTP-сервером и методы работы с SSL - сертификатами.

Также были получены навыки разработки приложений с помощью .NET Core Framework, а также навыки разработки WEB-приложений с использованием фреймворка Django на языке Python 3. Были получены навыки работы с PostgreSQL, навыки работы с протоколом TLS. Также были усовершенствованы навыки разработки программ с использованием объектно-ориентированного подхода.

Во время выполнения данной работы были разработаны: алгоритм загрузки данных, с использованием протокола TLS и SSL - сертификата, алгоритм загрузки данных на удаленный SFTP-сервер, а также веб-приложения для мониторинга полученных данных. Также была спроектирована БД для хранения полученных и обработанных данных.

					ПГУ 1.090301.09.001 ПЗ	Лист
						82
Изм.	Лист	№ докум.	Подпись	Дата		

Список использованных источников

1. Документация к фреймворку Django [Электронный ресурс] / Режим доступа: <https://www.djangoproject.com/start/> свободный (дата обращения 21.04.2025).
2. Документация к .NET Framework [Электронный ресурс] / Режим доступа: <https://learn.microsoft.com/en-us/dotnet/framework/> свободный (дата обращения 23.04.2025).
3. Документация к СУБД PostgreSQL [Электронный ресурс] / Режим доступа: <https://www.postgresql.org/docs/current/> свободный (дата обращения 22.04.2025).
4. Документация по работе с API АСОИ ФинЦЕРТ [Электронный ресурс] / Режим доступа: <https://lk.fincert.cbr.ru/> ограниченный (дата обращения 10.05.2025).
5. Доусон Майкл. Програмируем на Python. - 3-е изд. Прогресс книга, 2019. - 416 с.
6. Руководство по C# [Электронный ресурс] / Режим доступа: <https://learn.microsoft.com/ru-ru/dotnet/csharp/> свободный (дата обращения 23.04.2025).
7. Документация к SFTP-протоколу [Электронный ресурс] / Режим доступа: <https://sftpcloud.io/learn/sftp/sftp-commands-complete-guide> свободный (дата обращения 03.05.2025).

Приложение А

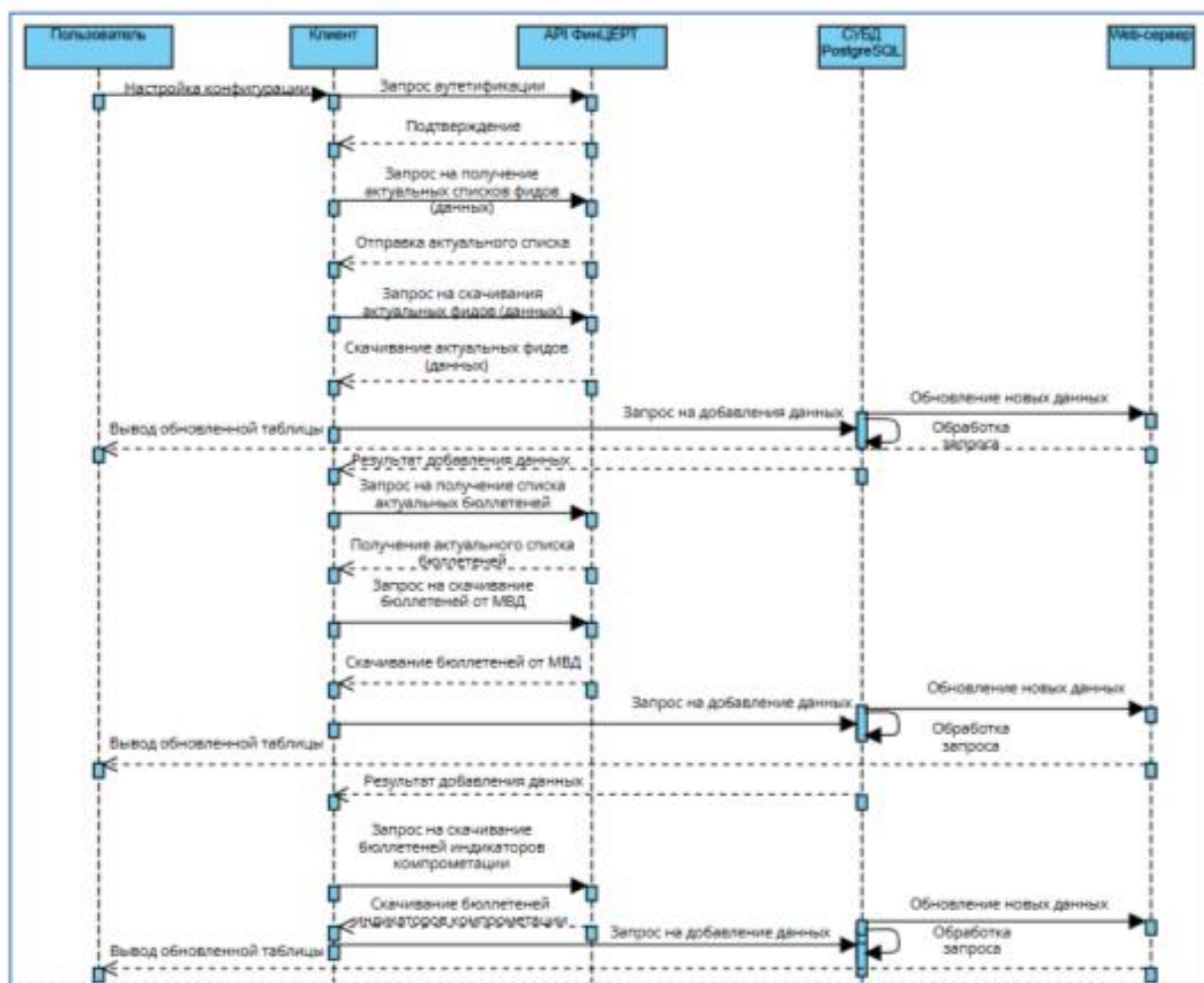
(Обязательное)

Модуль веб-приложения для автоматизированного сбора и систематизации
данных о неблагонадежных клиентах банка

UML – Диаграммы

					ПГУ 1.090301.09.001 ПЗ	Лист
						84
Изм.	Лист	№ докум.	Подпись	Дата		

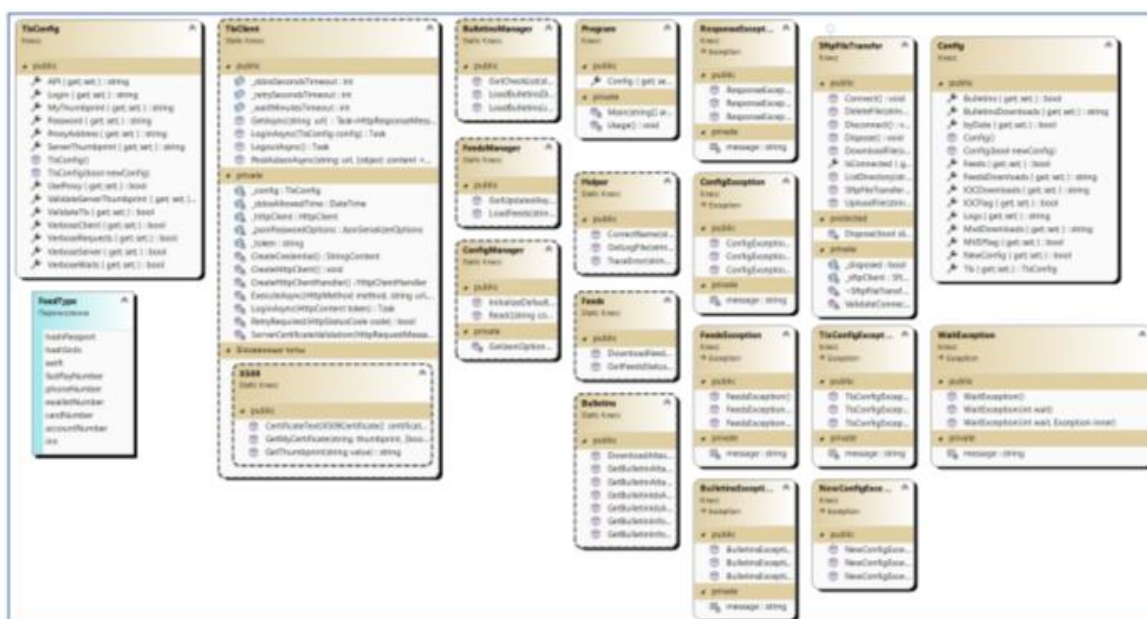
Приложение А.1 — Диаграмма последовательностей

[illegible]

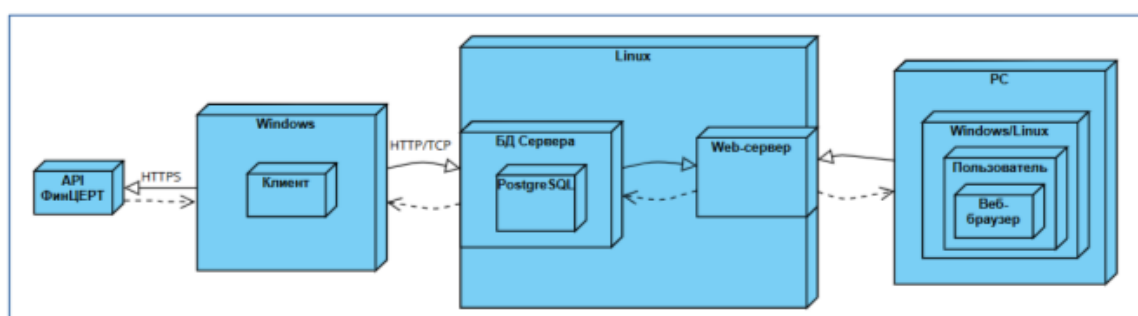
Приложение А.2 — Диаграмма компонентов

[illegible]

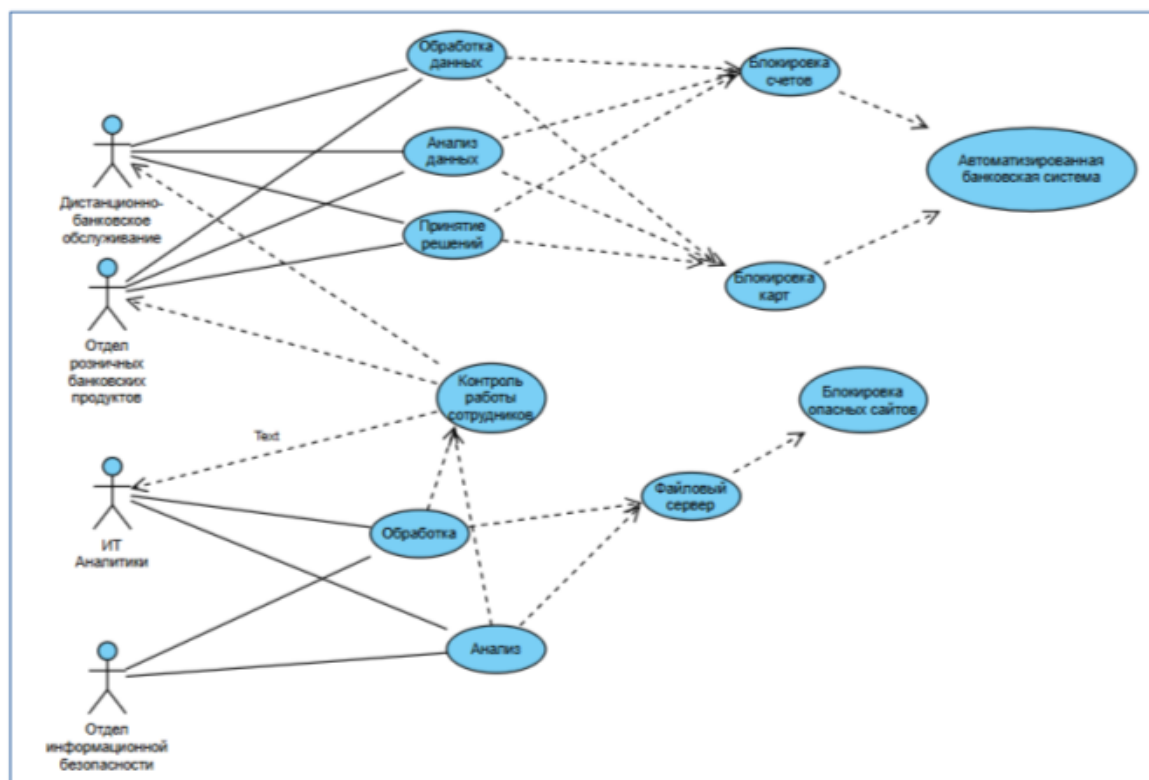
Приложение А.3 — Диаграмма классов

[illegible]

Приложение А.4 — Диаграмма развертывания

[illegible]

Приложение А.5 — Диаграмма вариантов использования



ПГУ 1.090301.09.002 Д13									
Изм.	Лист	№ докум.	Подпись	Дата	История изменений				
					№	Изм.	Дата	Исполнитель	Проверенный
					1				
					2				
					3				
					4				
					5				
					6				
					7				
					8				
					9				
					10				

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.09.001 ПЗ

Лист

89

Приложение Б

(Обязательное)

Модуль веб-приложения для автоматизированного сбора и
систематизации данных о неблагонадежных клиентах банка

Листинг программы

					ПГУ 1.090301.09.001 ПЗ	Лист
						90
Изм.	Лист	№ докум.	Подпись	Дата		

Файл Programm.cs:

```
using System.Diagnostics;
using API;
using FincertClient.Managers;
using TLS;
namespace FincertClient;
internal class Program
{
    public static Config Config { get; set; } = null!;
    static async Task<int> Main(string[] args)
    {
        try
        {
            // defaults
            bool checklist = true;
            bool feeds = true;
            int limit = 100;
            long offset = 0;
            string configType = null;
            Console.WriteLine("Hello, World!"); // :)

            Console.WriteLine("Подключение логирования на
консоль.");
            using ConsoleTraceListener ConTracer = new() { Name =
nameof(ConTracer) };
            Trace.Listeners.Add(ConTracer);

            // Опциональные параметры
            for (int i = 0; i < args.Length; i++)
            {
                string cmd = args[i].ToLower();

                switch (cmd)
                {
```

					ПГУ 1.090301.09.001 ПЗ	Лист
						91
Изм.	Лист	№ докум.	Подпись	Дата		

```

        case "-checklist":
            checklist = true;
            break;

        case "-feeds":
            feeds = true;
            break;

        case "-limit":
            if (args.Length > i)
                limit = int.Parse(args[++i]);
            break;

        case "-offset":
            if (args.Length > i)
                offset = long.Parse(args[++i]);
            break;

        case "-config":
            if (i + 1 < args.Length)
                configType = args[++i];
            break;

        default:
            if (!cmd.StartsWith("-"))
                configType = args[i]; // Поддержка
старого формата без -config
            else
                throw new
ArgumentException($"Непонятный параметр '{args[i]}'.");
            break;
    }
}

Trace.WriteLine("Чтение конфигурации...");

```

```

        Config = ConfigurationManager.Read(configType);
        var Password = Config.Tls.Password;
        if (OperatingSystem.IsWindows() &&
            string.IsNullOrEmpty(Password))
        {
            Trace.WriteLine($"Получение '{Login}' из
Диспетчера учетных данных Windows.");
            var cred =
CredentialManager.ReadCredential(Login);
            Config.Tls.Login = cred.UserName
                ?? throw new
CredManagerException(nameof(Login));
            Config.Tls.Password = cred.Password
                ?? throw new
CredManagerException(nameof>Password));
        }
    #endregion Config
    Trace.WriteLine("Login...");
    await TlsClient.LoginAsync(Tls!);
    Console.WriteLine(Config.IOCFlag);
    Console.WriteLine(Config.MVDFlag);
    // Скачать комплект файлов для чек-листа
    if (Config.MVDFlag || Config.IOCFlag)
    {
        if (checklist)
        {
            await BulletinsManager.GetCheckList(
                Path.Combine(Config.BulletinsDownloads,
"CheckList"), Config.byDate, Config.MVDFlag, Config.IOCFlag);
        }
    }
    if (Config.Feeds)
    {

```

```

        await
FeedsManager.LoadFeeds(Config.FeedsDownloads);

        goto exit

        await
BulletinsManager.LoadBulletinsDirs(Config.BulletinsDownloads,
limit, offset);
    }
    exit:
        Trace.WriteLine("Logout");
        await TlsClient.LogoutAsync();
        ConTracer.WriteLine("end.");
        FileTracer.WriteLine($"--- {DateTime.Now:T} end. ---
");

        return 0;
    }
    catch (Exception ex)
    {
        Helper.TraceError("Program Error", ex);
        return 1;
    }
    finally
    {
        Trace.Listeners.Clear();
    }
}

```

Файл Config.cs:

```

using TLS;
namespace FincertClient;
public class Config
{
    public bool NewConfig { get; set; }
    public TlsConfig? Tls { get; set; }
    public bool Feeds { get; set; }
    public string? FeedsDownloads { get; set; }
    public bool Bulletins { get; set; }
    public string? BulletinsDownloads { get; set; }
}

```

```

public bool byDate { get; set; }
public string? MvdDownloads { get; set; }
public string ? IOCDownloads { get; set; }
public string? Logs { get; set; }
public bool MVDFlag { get; set; }
public bool IOCFlag { get; set; }
public Config() { }
public Config(bool newConfig)
{
    NewConfig = newConfig;
    Tls = new(newConfig);
    Feeds = false;
    FeedsDownloads = nameof(FeedsDownloads);
    byDate = true;
    Bulletins = false;
    BulletinsDownloads = nameof(BulletinsDownloads);
    MvdDownloads = nameof(MvdDownloads);
    IOCDownloads = nameof(IOCDownloads);
    Logs = nameof(Logs);
    MVDFlag = false;
    IOCFlag = false;
}
}

```

Файл TlsClient.cs:

```

using System.Diagnostics;
using System.Net;
using System.Net.Http.Headers;
using System.Net.Http.Json;
using System.Net.Security;
using System.Reflection;
using System.Security.Authentication;
using System.Security.Cryptography.X509Certificates;
using System.Text;
using System.Text.Encodings.Web;
using System.Text.Json;

```

					ПГУ 1.090301.09.001 ПЗ	Лист
						95
Изм.	Лист	№ докум.	Подпись	Дата		

```

namespace TLS;

internal record Credential(string Login, string Password);

internal static class TlsClient
{
    private static HttpClient _httpClient = null!;
    private static TlsConfig _config = null!;
    private static string _token = string.Empty;
    private static readonly JsonSerializerOptions
_jsonPasswordOptions = new()
    { Encoder = JavaScriptEncoder.UnsafeRelaxedJsonEscaping };
    private static DateTime _ddosAllowedTime = DateTime.Now;
    public static readonly int _retrySecondsTimeout = 2; // retry
* RetryTimeout
    public static readonly int _ddosSecondsTimeout = 1;
    public static readonly int _waitMinutesTimeout = 10;
    public static async Task LoginAsync(TlsConfig config)
    {
        _config = config;
        var token = CreateCredential();
        CreateHttpClient();
        await LoginAsync(token);
    }
    private static StringContent CreateCredential()
    {
        string api = _config.API!;
        string login = _config.Login!;
        string password = _config.Password!;
        var cred = new Credential(login, password);
        string json = JsonSerializer.Serialize(cred,
_jsonPasswordOptions);
        var token = new StringContent(json, Encoding.UTF8,
"application/json");
        return token;
    }
    private static HttpClientHandler CreateHttpClientHandler()

```

```

{
    HttpClientHandler handler = new()
    {
        UseDefaultCredentials = false,
        ClientCertificateOptions =
ClientCertificateOption.Manual,},
        ServerCertificateCustomValidationCallback =
ServerCertificateValidation,
        SslProtocols = SslProtocols.Tls12
    };
    string thumbprint =
X509.GetThumbprint(_config.MyThumbprint!);
    X509Certificate2 certificate =
X509.GetMyCertificate(thumbprint);
    handler.ClientCertificates.Add(certificate);
    if (_config.VerboseClient)
    {
        Console.WriteLine("--- Client ---");
        Console.WriteLine(X509.CertificateText(certificate));
    }
    if (_config.UseProxy)
    {
        // DefaultProxyCredentials = null;
        handler.UseProxy = true;
        handler.Proxy = new WebProxy(_config.ProxyAddress);
    }
    return handler;
}

private static bool
ServerCertificateValidation(HttpRequestMessage requestMessage,
    X509Certificate2? certificate, X509Chain? chain,
SslPolicyErrors sslErrors)
{
    if (certificate is null)
        return false;

```

```

        if (_config.VerboseServer)
        {
            Console.WriteLine("--- Server ---");
            Console.WriteLine($"Request:
{requestMessage.RequestUri}");
            Console.WriteLine(X509.CertificateText(certificate));
            Console.WriteLine($"Tls errors: {sslErrors}");
        }
        if (_config.ValidateTls && ValidateTlsFail(sslErrors))
            return false;
        if (_config.ValidateServerThumbprint &&
ValidateServerThumbprintFail(certificate))
            return false;
        return true;
        static bool ValidateServerThumbprintFail(X509Certificate2
certificate)
        {
            bool fail = certificate.GetCertHashString() !=
                X509.GetThumbprint(_config.ServerThumbprint!);
            Trace.WriteLineIf(fail, "ServerThumbprint not match
this server.");
            return fail;
        }
        static bool ValidateTlsFail(SslPolicyErrors sslErrors)
        {
            bool fail = sslErrors != SslPolicyErrors.None;
            Trace.WriteLineIf(fail, "TLS protocol error.");
            return fail;
        }
    }
    private static void CreateHttpClient()
    {
        var handler = CreateHttpClientHandler();
        _httpClient = new(handler, true);
    }

```

```

        string app =
Path.GetFileNameWithoutExtension(Environment.ProcessPath)
        ?? nameof(TlsClient);
        string ver =
Assembly.GetEntryAssembly()?.GetName().Version?.ToString()
        ?? "8.0";
        _httpClient.DefaultRequestHeaders.UserAgent.Add(
            new ProductInfoHeaderValue(app, ver));
        MediaTypeWithQualityHeaderValue("application/json"));
        private static async Task LoginAsync(HttpContent token)
        {
            var response = await ExecuteAsync(HttpMethod.Post,
"account/login", token);
            _token = await response.Content.ReadAsStringAsync();
            _httpClient.DefaultRequestHeaders.Authorization =
new("Bearer", _token);
            public static async Task LogoutAsync()
            {
                await ExecuteAsync(HttpMethod.Post, "account/logout");
                _token = string.Empty;
                _httpClient.DefaultRequestHeaders.Authorization =
new("Bearer", _token);
                _httpClient.Dispose();
            }
            while (DateTime.Now < end)
            {
                if (DateTime.Now < _ddosAllowedTime)
                    Thread.Sleep(_ddosAllowedTime - DateTime.Now);
                using (var request = new HttpRequestMessage(method,
api))
                {
                    if (content != null)
                        request.Content = content;

                    retry++;
                }
            }
        }
    }
}

```

```

        var response = await
_httpClient.SendAsync(request);
        _ddosAllowedTime =
DateTime.Now.AddSeconds(_ddosSecondsTimeout);
        code = response.StatusCode;
        if (!RetryRequired(code))
        {
            if (code == HttpStatusCode.OK) // for this API
only!

                return response;
            else
                throw new ResponseException(code);
        }
        }
        int pause = retry * _retrySecondsTimeout;
        Trace.WriteLineIf(_config.VerboseWaits,
            $"Код {code} - повтор {retry} через {pause}
сек.");
        Thread.Sleep(pause * 1000);
    }
    throw new WaitException(_waitMinutesTimeout);
}
private static bool RetryRequired(HttpStatusCode code)
{
    internal static class X509
    {
        public static string GetThumbprint(string value)
            => value.Replace(" ", string.Empty).ToUpper();
        public static X509Certificate2 GetMyCertificate(string
thumbprint, bool validOnly = true)
        {
            if (string.IsNullOrEmpty(thumbprint))
                throw new Exception(
                    "My certificate thumbprint not found in
config.");

```

```

        using X509Store store = new(StoreName.My,
StoreLocation.CurrentUser, OpenFlags.ReadOnly);

        var found =
store.Certificates.Find(X509FindType.FindByThumbprint, thumbprint,
validOnly);

        if (found.Count == 1)
            return found[0];
        throw new Exception(
            $"My certificate with thumbprint '{thumbprint}'
not found in storage.");

        public static string CertificateText(X509Certificate2?
certificate)
        {
            if (certificate is null)
                return "Certificate not found.";
            StringBuilder sb = new();
            return sb
                .AppendLine($"From date:
{certificate.GetEffectiveDateString()}")
                .AppendLine($"Exp date:
{certificate.GetExpirationDateString()}")
                .AppendLine($"Issuer:      {certificate.Issuer}")
                .AppendLine($"Subject:    {certificate.Subject}")
                .AppendLine($"Serial num:
{certificate.GetSerialNumberString()}") //SerialNumber
                .AppendLine($"Thumbprint:
{certificate.GetCertHashString()}") //Thumbprint
                .ToString();
        }
    }
}

```

Файл TlsConfig.cs:

```

namespace TLS;

public class TlsConfig
{

```

					ПГУ 1.090301.09.001 ПЗ	Лист
						101
Изм.	Лист	№ докум.	Подпись	Дата		

```

public bool VerboseClient { get; set; }
public string? MyThumbprint { get; set; }
public string? Login { get; set; }
public string? Password { get; set; }
public bool VerboseServer { get; set; }
public string? API { get; set; } ServerThumbprint).
public bool ValidateServerThumbprint { get; set; }
public string? ServerThumbprint { get; set; }
public bool ValidateTls { get; set; }
public bool UseProxy { get; set; }
public string? ProxyAddress { get; set; }
public bool VerboseRequests { get; set; }
public bool VerboseWaits { get; set; }
public TlsConfig() { }
public TlsConfig(bool newConfig)
{
    VerboseClient = newConfig;
    MyThumbprint = string.Empty;
    Login = "FinCERT";
    Password = string.Empty;
    VerboseServer = false;
    API = "https://zoe-api.fincert.cbr.ru/api/v1/";
    ValidateServerThumbprint = false;
    ServerThumbprint = string.Empty;
    ValidateTls = false;
    UseProxy = false;
    ProxyAddress = "http://192.168.2.1:3128";
    VerboseRequests = false;
    VerboseWaits = false;
}
}

```

Файл BulletinsManager.cs:

```

using System.Diagnostics;
using System.IO.Compression;
using System.Text;

```

					ПГУ 1.090301.09.001 ПЗ	Лист
						102
Изм.	Лист	№ докум.	Подпись	Дата		

```

using API;
using static API.Bulletins;
using static FincertClient.Managers.Helper;
using FincertClient.DatebaseManager;
using IOCFeeds;
using Npgsql;
namespace FincertClient.Managers;
internal static class BulletinsManager
{
    public static async Task GetCheckList(string path, bool byDate
= false, bool MVDFlag = false, bool IOCFlag = false, int limit =
20)
    {
        ids2[i] = ids.Items[i].Id;
        if (ids2[i].Contains(idtoday))
        {
            IDResultToday[i] = ids2[i];
        }
    }

    if (byDate == true)
    {
        await GetBulletinInfosAsync(IDResultToday,
Path.Combine(path, "2-ids-infos.json"));
        var id = IDResultToday[0];
        await GetBulletinAttachInfoAsync(id,
Path.Combine(path, "3-id-info.json"));
        await LoadBulletinsDirs(path, MVDFlag, IOCFlag,
limit);
    } else
    {
        await GetBulletinInfosAsync(ids2, Path.Combine(path,
"2-ids-infos.json"));
        var id = ids2[0];
    }
}

```

```

        await GetBulletinAttachInfoAsync(id,
Path.Combine(path, "3-id-info.json"));

        await LoadBulletinsDirs(path, MVDFlag, IOCFlag,
limit);
    }
}

public static async Task LoadBulletinsList(string path, int
limit = 100, long offset = 0)
{
    Directory.CreateDirectory(path);
    Trace.WriteLine("Получение бюллетеней списком...");
    var ids = await GetBulletinIdsAsync(limit, offset);
    int count = ids.Items.Length;
    string[] ids2 = new string[count];
    Trace.WriteLine($"Всего: {ids.Total}, в листе: {count}.");
    for (int i = 0; i < count; i++)
        ids2[i] = ids.Items[i].Id;
    var BulletinInfos = await GetBulletinInfosAsync(ids2);
    StringBuilder sb = new();
    foreach (var bulletin in BulletinInfos.Items)
    {
        var id = bulletin.Id;
        var date = DateTime.Parse(bulletin.PublishedDate);
        sb.AppendLine(new string('-', 36))
            .AppendLine(id)
            .AppendLine($"Дата публикации: {date:g}")
            .AppendLine($"Идентификатор: {bulletin.Hrid}")
            .AppendLine($"Заголовок: {bulletin.Header}")
            .AppendLine($"Тип рассылки: {bulletin.Type}")
            .AppendLine($"Подтип рассылки:
{bulletin.Subtype}");

        if (bulletin.AttachmentId != null)
            sb.AppendLine($"Файл рассылки:
{bulletin.AttachmentId}");
    }
}

```

```

        sb.AppendLine($"Описание:
{bulletin.Description}");
    }
    string text = sb.ToString();
    string file = Path.Combine(path, "Bulletins.txt");
    await File.WriteAllTextAsync(file, text);
    Console.WriteLine(text);
}

public static async Task LoadBulletinsDirs(string path, bool
MVDFlag, bool IOCFlag, int limit = 100, long offset = 0)
{

    foreach (var itemId in ids.Items)
    {
        var id = itemId.Id;
        var bulletin = await GetBulletinAttachInfoAsync(id);
        if (!bulletin.Hrid.Contains(currentDate))
        {
            Trace.WriteLine($"Пропущен бюллетень
{bulletin.Hrid} - не соответствует текущей дате");
            continue;
        }
        var date = DateTime.Parse(bulletin!.PublishedDate);
        var name = CorrectName($"{bulletin.Hrid.Trim()}");
        var dir = Path.Combine(path, name)
        StringBuilder sb = new();
        sb.AppendLine(id)
            .AppendLine($"Дата публикации: {date:g}")
            .AppendLine($"Идентификатор: {bulletin.Hrid}")
            .AppendLine($"Заголовок: {bulletin.Header}")
            .AppendLine($"Тип рассылки: {bulletin.Type}")
            .AppendLine($"Подтип рассылки:
{bulletin.Subtype}");
        if (bulletin.Attachment != null)
        {

```

```

        string filename = bulletin.Attachment.Name;
        sb.AppendLine($"Файл рассылки: {filename}");
        string attach = Path.Combine(dir, filename);
        if (filename.StartsWith("feeds_20",
StringComparison.OrdinalIgnoreCase) && MVDFlag == true)
        {
            if (Directory.Exists(dir))
            {
                if (offset == 0)
                {
                    Trace.WriteLine($"{name} // конец скачивания.");
                    break;
                }
                else
                {
                    Trace.WriteLine($"{name} // есть");
                    continue;
                }
            }
            Trace.WriteLine(name);
            Directory.CreateDirectory(dir);
            await
DownloadAttachmentAsync(bulletin.Attachment.Id, attach);
            if (string.Compare(filename, lastMvdZip, true)
> 0) //TODO
            {
                Trace.WriteLine($"Получен фид МВД
'{filename}'");
                lastMvdZip = filename;
                string dir1 = Program.Config.MvdDownloads!; //TODO
                if (!dir1.Equals(string.Empty))
                    Directory.CreateDirectory(dir1);
                ZipFile.ExtractToDirectory(attach, dir1, true);
            }
        }
    }
}

```

```

        var connector = new
PGSQLConnector("Host=localhost;Database=users;Username=postgres;Pa
ssword=shkola77dan");

        connector.LoadMvdTypeTable(bulletin.Hrid,
currentDate);

    }

}

    if (filename.StartsWith("FinCERT_20",
StringComparison.OrdinalIgnoreCase) && IOCFlag == true)
    {
        if (Directory.Exists(dir))
        {
            if (offset == 0)
            {
                Trace.WriteLine($"{name} // конец
скачивания.");

                break;
            }
            else
            {
                Trace.WriteLine($"{name} // есть");
                continue;
            }
        }
        Trace.WriteLine(name);
        Directory.CreateDirectory(dir);
        await
DownloadAttachmentAsync(bulletin.Attachment.Id, attach);
        if (string.Compare(filename, lastIocZip, true)
> 0)

        {
            Trace.WriteLine($"Получен IOC файл
'{filename}'");
            lastIocZip = filename;
            string iocDir = Program.Config.IOCDownloads!;

```

					ПГУ 1.090301.09.001 ПЗ	Лист
						107
Изм.	Лист	№ докум.	Подпись	Дата		

```

if (!string.IsNullOrEmpty(iocDir))
{
    Directory.CreateDirectory(iocDir);
    string destinationPath = Path.Combine(iocDir, filename);
    File.Copy(attach, destinationPath, overwrite: true);
    try
    {
        var results = processor.ProcessLatestArchive("C:\\\\IOCDownloads");
        processor.SaveResultsToFile(results, @"C:\\IOCDownloads\\ExtractDataToChek.csv");
        Trace.WriteLine($"Обработано {results.Count} записей");
        ipChecker.ProcessBlockLists( @"C:\\IOCDownloads\\Network FinCert Blocked IP.txt", @"C:\\IOCDownloads\\ExtractDataToChek.csv");
        using var sftp = new SftpFileTransfer("localhost", 22, "karev", "C:\\\\Users\\\\karev\\.ssh\\id_ed25519.ppk");
        sftp.Connect();
        if (sftp.IsConnected)
        {
            Trace.WriteLine("Подключение к SFTP серверу успешно!");
            sftp.UploadFile("C:\\\\IOCDownloads\\Network FinCert Blocked IP.zip", "Network FinCert Blocked IP.zip");
            Trace.WriteLine("Файл успешно загружен");
            sftp.UploadFile("C:\\\\IOCDownloads\\version.txt", "version.txt");
            Trace.WriteLine("Файл успешно загружен");
        }
        var connector = new
        PGSQLConnector("Host=localhost;Database=users;Username=postgres;Password=shkola77dan");
        var dayPublished = date.ToString();
        connector.LoadIOCDateDownloads(bulletin.Hrid, currentDate, dayPublished);
    }
}

catch (Exception ex)
{
    Trace.WriteLine($"Ошибка: {ex.Message}");
    if (MVDFlag == true)

```

```

    {
        var connector = new
PGSQLConnector("Host=localhost;Database=users;Username=postgres;Pa
ssword=shkola77dan");

        connector.ExportToCsvFromConfig("D:\\FinCERT-
Client\\FinCERT-Client-main\\FinCERT-Client-main\\FinCERT-
Client\\bin\\Debug\\net8.0\\FinCertMVDFeedsConfigs.json");
    }
}
}

```

Файл PGSQLConnector.cs:

```

using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.IO;
using System.Net.Http.Json;
using System.Security.Cryptography;
using System.Text.Json.Serialization;
using Newtonsoft.Json;
using Npgsql;
namespace FincertClient.DatebaseManager;
    public class PGSQLConnector
    {
        private readonly string _connectionString;
        public PGSQLConnector(string connectionString)
        {
            _connectionString = connectionString;
        }
        public void ExportToCsvFromConfig(string jsonConfigPath)
        {
            var configs = LoadImportConfigs(jsonConfigPath);
            ProcessMultipleTables(configs);
        }
        public void LoadMvdTypeTable(string MvdHrid, string
currentDate)

```

					ПГУ 1.090301.09.001 ПЗ	Лист
						109
Изм.	Лист	№ докум.	Подпись	Дата		

```

        {
            using (var connection = new
NpgsqlConnection(_connectionString))
            {
                connection.Open();
                var insertMvdCommand = $"INSERT INTO
public.users_FeedsMvdDateDownloads (date, ""nameFeedsTrim"",
feedstype_id)
VALUES ('{currentDate}', '{MvdHrid}', 2)";
                using (var cmd = new NpgsqlCommand(insertMvdCommand, connection))
                {
                    cmd.ExecuteNonQuery();
                }
            }
        }

        public void LoadFinCertDateDownloads(string currentDate)
        {
            using (var connection = new
NpgsqlConnection(_connectionString))
            {
                connection.Open();
                var insertFinCertDateCommand = $"INSERT INTO
public.users_Feedsfincertdatedownloads (date, feedstype_id)
VALUES ('{currentDate}',
1)";
                using (var cmd = new
NpgsqlCommand(insertFinCertDateCommand, connection))
                {
                    cmd.ExecuteNonQuery();
                }
            }
        }

        public void LoadIOCDDateDownloads(string IOCHrid, string
currentDate, string dateMailing)
        {
            using (var connection = new
NpgsqlConnection(_connectionString))

```

```

        {
            connection.Open();
            var insertIOCDateCommand = $"INSERT INTO
public.users_fincertiocdateprocessing ("dateMailing",
"ioctrim", feedstype_id, date)
VALUES ('{dateMailing}', '{IOCHrid}', 3, '{currentDate}')";
            using (var cmd = new
NpgsqlCommand(insertIOCDateCommand, connection))
            {
                cmd.ExecuteNonQuery();
            }
            public List<TableImportConfig> LoadImportConfigs(string
jsonConfigPath)
            {
                if (!File.Exists(jsonConfigPath))
                    throw new FileNotFoundException($"Конфигурационный
файл не найден: {jsonConfigPath}");

                var json = File.ReadAllText(jsonConfigPath);
                return
JsonConvert.DeserializeObject<List<TableImportConfig>>(json);
            }
            public void ProcessMultipleTables(List<TableImportConfig>
importConfigs)
            {
                using (var connection = new
NpgsqlConnection(_connectionString))
                {
                    connection.Open();
                    foreach (var config in importConfigs)
                    {
                        if (!File.Exists(config.CsvFilePath))
                        {
                            Console.WriteLine($"Файл не найден:
{config.CsvFilePath}");

```

```

        continue;
    }

    ProcessSingleTable(connection, config);

    private void ProcessSingleTable(NpgsqlConnection
connection, TableImportConfig config)
    {
        Trace.WriteLine("-----Начало импорта в БД-----
----");

        Trace.WriteLine($"\\nОбработка таблицы:
{config.TableName}");

        Trace.WriteLine($"Файл:
{Path.GetFileName(config.CsvFilePath)}");

        var tempTableName = $"public_temp{config.TableName}";

        try
        {
            var copyCommand =
ImportToDatabaseFromCsvFilesBach(connection, tempTableName,
config);

            using (var cmd = new NpgsqlCommand(copyCommand, connection))
            {
                cmd.ExecuteNonQuery();
            }
        }
        finally
        {
            \\DropTempTable(connection, tempTableName);
            Trace.WriteLine("-----Импорт в БД успешен-----");
        }
    }

    private string
ImportToDatabaseFromCsvFilesBach(NpgsqlConnection connection,
string tempTableName, TableImportConfig config)
    {
        var columns = new List<string>();

        foreach (var mapping in config.ColumnMappings)

```

```

        var columnType = mapping.Value == "count" ? "INTEGER"
: "TEXT";

        columns.Add($"{mapping.Value}\" {columnType}");
    }

    var createTempTable = $"CREATE TEMP TABLE {tempTableName}
({string.Join(", ", columns)});";

    var csvColumns =
config.ColumnMappings.Keys; //.Concat(config.KeyColumns ??
Enumerable.Empty<string>());

    var dbColumns =
config.ColumnMappings.Values; //.Concat(config.KeyColumns ??
Enumerable.Empty<string>());

    var importToTempTable = @$"COPY
{tempTableName} ("'{string.Join("\", \'", dbColumns)}'")
        FROM '{config.CsvFilePath}'
        DELIMITER ','
        CSV HEADER
        ENCODING 'UTF-8';";

    if (config.KeyColumns == null ||
!config.KeyColumns.Any())
        throw new ArgumentException("Не указаны ключевые
колонки для проверки дубликатов");

    var joinConditions = config.KeyColumns
        .Select(k => $"t.\"{k}\" = s.\"{k}\"");
    var insertColumns = config.ColumnMappings.Values
        .Select(c => $"\"{c}\"");
    var selectValues = config.ColumnMappings.Values
        .Select(c => $"s.\"{c}\"");
    var insertNewRecords = @$"
        INSERT INTO public.\"{config.TableName}\"
({string.Join(", ", insertColumns)})
        SELECT {string.Join(", ", selectValues)}
        FROM {tempTableName} s
        WHERE NOT EXISTS(
            SELECT 1

```

```

        FROM public.""{config.TableName}"" t
        WHERE {string.Join(" AND ", joinConditions)}};";

        var updateDateExcludes = $"{UPDATE
public.""{config.TableName}"" t
    SET ""dateExcluded"" = CURRENT_DATE
    WHERE NOT EXISTS (
        SELECT 1
        FROM {tempTableName} s
        WHERE {string.Join(" AND ", joinConditions)}}
        AND t.""dateExcluded"" is NULL;";

        var bachCommand = createTempTable + "\n" +
importToTempTable + "\n" + insertNewRecords + "\n" +
updateDateExcludes;

        return bachCommand;
    }

    private void DropTempTable(NpgsqlConnection connection,
string tempTableName)
    {
        using (var cmd = new NpgsqlCommand($"DROP TABLE IF
EXISTS {tempTableName};", connection))
        {
            cmd.ExecuteNonQuery();
        }
    }

    public class TableImportConfig
    {
        public string TableName { get; set; }
        public string CsvFilePath { get; set; }
        public Dictionary<string, string> ColumnMappings { get;
set; }
        public List<string> KeyColumns { get; set; }
        public string Encoding { get; set; } = "UTF-8";
    }

```

Файл Feeds.cs:

```

using System.Net.Http.Json;
using TLS;

```

					ПГУ 1.090301.09.001 ПЗ	Лист
						114
Изм.	Лист	№ докум.	Подпись	Дата		

```

namespace API;

public record FeedsStatus(
    string UploadDatetime,
    string Type,
    int Version);

internal static class Feeds
{
    public static async Task<FeedsStatus>
GetFeedsStatusAsync(FeedType feed)
    {
        string url = $"antifraud/feeds/{feed}";
        using var response = await TlsClient.GetAsync(url);
        response.EnsureSuccessStatusCode();
        return await
response.Content.ReadFromJsonAsync<FeedsStatus>()
            ?? throw new FeedsException("Статус фидов не
получен.");
    }

    public static async Task DownloadFeedsAsync(FeedType feed,
string path)
    {
        Directory.CreateDirectory(path);
        string url = $"antifraud/feeds/{feed}/download";
        using var response = await TlsClient.GetAsync(url);
        response.EnsureSuccessStatusCode();
        string name = feed switch
        {
            FeedType.accountNumber => "account_number",
            FeedType.cardNumber     => "card_number",
            FeedType.ewalletNumber  => "ewallet_number",
            FeedType.fastPayNumber  => "fastpay_number",
            FeedType.hashPassport   => "passport_hash",
            FeedType.hashSnils      => "snils_hash",
            FeedType.inn            => "inn",
        }
    }
}

```

```

        FeedType.phoneNumber    => "phone_number",
        FeedType.swift          => "swift",
        _ => feed.ToString()
    };

```

```
public class FeedsException : Exception
```

Файл admin.py:

```

from django.contrib import admin
from .models import (
    FeedsTypeNavigation,
    FeedsFinCertDateDownloads,
    FinCertIocDateProcessing
)
admin.site.register(FeedsTypeNavigation)
admin.site.register(FeedsFinCertDateDownloads)
admin.site.register(FinCertIocDateProcessing)

```

Файл models.py:

```

from django.db import models
#Таблица для навигации
class FeedsTypeNavigation(models.Model):
    feeds_name = models.CharField('Название фидов', max_length=20)
    feeds_description = models.TextField('Описание фидов')
    class Meta:
        verbose_name = 'Типы фидов (данных)'
        verbose_name_plural = verbose_name
class FeedsFinCertDateDownloads(models.Model):
    feedstype = models.ForeignKey(FeedsTypeNavigation,
on_delete=models.CASCADE)
    date = models.CharField('Время скачивания
фидов',max_length=30)
class FeedsEwalletNumbers(models.Model):
    datedownloads = models.ForeignKey(FeedsFinCertDateDownloads,
on_delete=models.SET_NULL, null=True)
    ewalletNumbers = models.CharField('Номер кошелька',
max_length=64)
    paymentSystem = models.CharField('Название платежной системы',

```

```

max_length=100)
    dateFixed = models.CharField('Дата и время фиксации
инцидента', max_length=30)
    count = models.SmallIntegerField('Количество')
    country = models.CharField('Страна', max_length=15)
    dateExcluded = models.DateTimeField("Дата удаления из списка",
null=True)
class FeedsPhoneNumbers(models.Model):
    datedownloads = models.ForeignKey(FeedsFinCertDateDownloads,
on_delete=models.SET_NULL, null=True)
    phoneNumbers = models.CharField('Номер телефона',
max_length=64)
    dateFixed = models.CharField('Дата и время фиксации
инцидента', max_length=30)
    count = models.SmallIntegerField('Количество')
    country = models.CharField('Страна', max_length=15)
    dateExcluded = models.DateTimeField("Дата удаления из списка",
null=True)
class FeedsSnilsHash(models.Model):
    datedownloads = models.ForeignKey(FeedsFinCertDateDownloads,
on_delete=models.SET_NULL, null=True)
    hashSnils = models.CharField('Хеш СНИЛСа', max_length=200)
    dateFixed = models.CharField('Дата и время фиксации
инцидента', max_length=30)
    count = models.SmallIntegerField('Количество')
    country = models.CharField('Страна', max_length=15)
    dateExcluded = models.DateTimeField("Дата удаления из списка",
null=True)

class FeedsSwift(models.Model):
    datedownloads = models.ForeignKey(FeedsFinCertDateDownloads,
on_delete=models.SET_NULL, null=True)
    accountNumbersSwift = models.CharField('Номер счета',
max_length=150)
    swiftBIC= models.CharField('SWIFT БИК', max_length=50)

```

					ПГУ 1.090301.09.001 ПЗ	Лист
						117
Изм.	Лист	№ докум.	Подпись	Дата		

```

        dateFixed = models.CharField('Дата и время фиксации
инцидента', max_length=30)
        count = models.SmallIntegerField('Количество')
        country = models.CharField('Страна', max_length=15)
        dateExcluded = models.DateTimeField("Дата удаления из списка",
null=True)
class FeedsMvdCardNumbers(models.Model):
    datedownloads = models.ForeignKey(FeedsMvdDateDownloads,
on_delete=models.SET_NULL, null=True)
    cardNumbers = models.CharField('Номер карты', max_length=64)
    dateExcluded = models.DateTimeField("Дата удаления из списка",
null=True)
class FeedsMvdFastPayNumbers(models.Model):
    datedownloads = models.ForeignKey(FeedsMvdDateDownloads,
on_delete=models.SET_NULL, null=True)
    fastpayNumber = models.CharField('Номер телефона',
max_length=64)
    dateExcluded = models.DateTimeField("Дата удаления из списка",
null=True)
class FeedsMvdInn(models.Model):
    datedownloads = models.ForeignKey(FeedsMvdDateDownloads,
on_delete=models.SET_NULL, null=True)
    inn = models.CharField('ИНН', max_length=30)
    dateExcluded = models.DateTimeField("Дата удаления из списка",
null=True)
class FeedsMvdPassportHash(models.Model):
    datedownloads = models.ForeignKey(FeedsMvdDateDownloads,
on_delete=models.SET_NULL, null=True)
    passportHash = models.CharField('Хеш паспорта',
max_length=150)
    dateExcluded = models.DateTimeField("Дата удаления из списка",
null=True)

```

Файл urls.py:

```

from django.urls import path
from . import views

```

					ПГУ 1.090301.09.001 ПЗ	Лист
						118
Изм.	Лист	№ докум.	Подпись	Дата		

```
urlpatterns = [
    path('', views.unified_feeds_view, name='unified-feeds'),
]
```

Файл views.py:

```
from django.core.paginator import Paginator
from django.shortcuts import render
from .models import *
from django.db.models import Q

def unified_feeds_view(request):
    items_per_page = 50
    active_tab = request.GET.get('tab', 'fincert')
    date_filter = request.GET.get('date_filter', '')
    fincert_dates =
FeedsFinCertDateDownloads.objects.select_related('feedstype').orde
r_by('-date')[:10]
    def get_filtered_queryset(model, date_field='dateFixed'):
        queryset =
model.objects.select_related('datedownloads').all()
        if date_filter:
            queryset =
queryset.filter(**{f'{date_field}__icontains': date_filter})
        return queryset.order_by(f'-{date_field}')
    account_numbers = get_filtered_queryset(FeedsAccountNumbers)
    card_numbers = get_filtered_queryset(FeedsCardNumbers)
    ewallet_numbers = get_filtered_queryset(FeedsEwalletNumbers)
    mvd_dates = FeedsMvdDateDownloads.objects.all().order_by('-
date')[:10]
    mvd_accounts = get_filtered_queryset(FeedsMvdAccountNumbers,
'datedownloads__date')
    mvd_cards = get_filtered_queryset(FeedsMvdCardNumbers,
'datedownloads__date')
    mvd_fastpay = get_filtered_queryset(FeedsMvdFastPayNumbers,
'datedownloads__date')
    mvd_inn = get_filtered_queryset(FeedsMvdInn,
'datedownloads__date')
```

					ПГУ 1.090301.09.001 ПЗ	Лист
						119
Изм.	Лист	№ докум.	Подпись	Дата		

```

        mvd_passports = get_filtered_queryset(FeedsMvdPassportHash,
        'datedownloads__date')
        ioc_data =
FinCertIocDateProcessing.objects.select_related('feedstype').order
_by('-dateMailing')[:10]
        def paginate_data(request, data, param_name):
            paginator = Paginator(data, items_per_page)
            page_number = request.GET.get(f'{param_name}_page')
            return paginator.get_page(page_number)
        context = {
            'active_tab': active_tab,
            'date_filter': date_filter,
            'fincert_dates': fincert_dates,
            'account_numbers_page': paginate_data(request,
account_numbers, 'account'),
            'card_numbers_page': paginate_data(request, card_numbers,
'card'),
            'ewallet_numbers_page': paginate_data(request,
ewallet_numbers, 'ewallet'),
            'fastpay_numbers_page': paginate_data(request,
fastpay_numbers, 'fastpay
            'mvd_dates': mvd_dates,
            'mvd_accounts': mvd_accounts,
            'mvd_cards': mvd_cards,
            'mvd_fastpay': mvd_fastpay,
            'mvd_inn': mvd_inn,
            'mvd_passports': mvd_passports,
            'ioc_data': ioc_data,
        }
    return render(request, 'users/unified_view.html', context)

```

Файл manage.py:

```

import os
import sys
def main():

```

					ПГУ 1.090301.09.001 ПЗ	Лист
						120
Изм.	Лист	№ докум.	Подпись	Дата		

```

    """Run administrative tasks."""
    os.environ.setdefault('DJANGO_SETTINGS_MODULE',
'prdip.settings')
    try:
        from django.core.management import
execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed
and "
            "available on your PYTHONPATH environment variable?
Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)

if __name__ == '__main__':
    main()

```

					ПГУ 1.090301.09.001 ПЗ	Лист
						121
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение В

(Обязательное)

Модуль веб-приложения для автоматизированного сбора и систематизации
данных о неблагонадежных клиентах банка

Презентация

					ПГУ 1.090301.09.001 ПЗ	Лист
						122
Изм.	Лист	№ докум.	Подпись	Дата		

Модуль веб-приложения для автоматизированного сбора и систематизации данных о неблагонадежных клиентах банка

Автор работы: студент группы 21BBB1

Карев Д.Ф.

Научный руководитель: к.э.н. доцент кафедры ВТ

Акифьев И. В.

Актуальность темы



- Сегодня банки и финансовые компании сталкиваются с серьезной проблемой - им нужно постоянно проверять клиентов на возможную причастность к мошенническим схемам.
- Каждый день в банк приходят новые клиенты, совершаются тысячи операций. Сотрудникам нужно проверять всех и вся по специальным спискам, которые к тому же регулярно обновляются. Если что-то пропустить - банк может получить крупный штраф или даже лишиться лицензии.
- Центральный Банк России создал специальный сервис - API ФинЦерт. Это как "окно" в базу данных всех подозрительных лиц и организаций. Но чтобы этим удобно пользоваться, нужно специальное программное обеспечение.

Цель работы

Целью данной работы является разработка программного обеспечения, которое должно помочь банкам быстрее проверять клиентов, снизить вероятность ошибок.

Объектом работы является модуль веб-приложения, выполняющий следующие функции:

- автоматически получать свежие данные из ФинЦерт;
- аккуратно хранить их в базе данных.

Изм.	Лист	№ докум.	Подпись	Дата

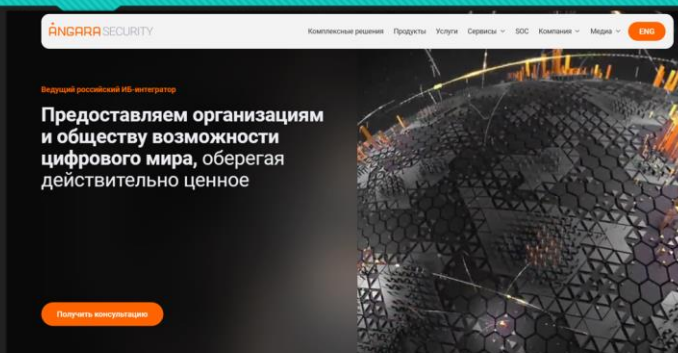
В процессе разработки должны будут решены следующие задачи

- Реализовать безопасное подключение к API ФинЦЕРТ с использованием TLS-сертификатов.
- Автоматизировать процесс получения свежих данных из ФинЦерт
- Разработка веб-интерфейса для мониторинга информации.
- Интеграция с SFTP-сервером для обмена данными.
- Создание базы данных для хранения и систематизации полученных данных.

Аналоги

- "Angara Security"
- "Fraudwall"

Сайт "Angara Security"



Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.09.001 ПЗ

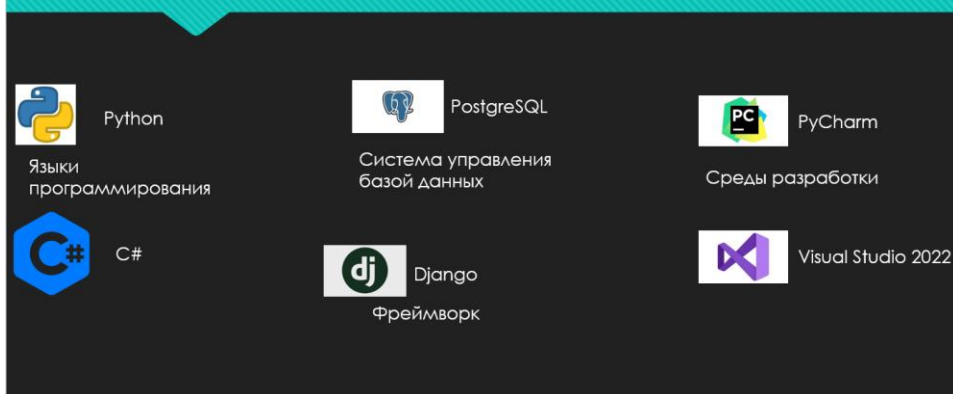
Лист

124

Сайт "Fraudwall"

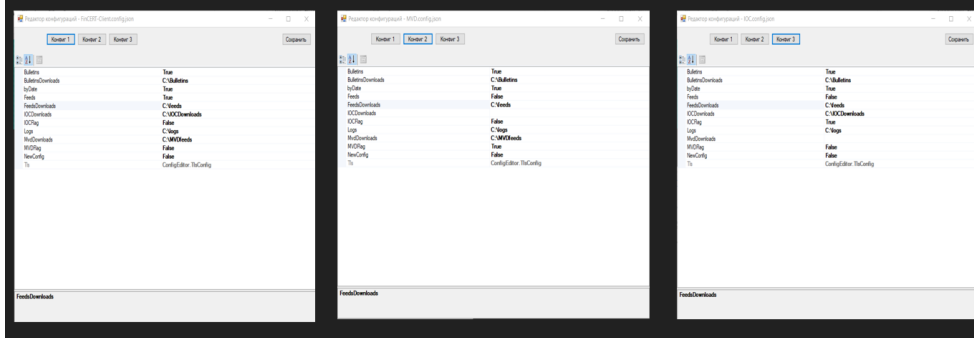


Стек технологий



Графический интерфейс

Использовалась технология Windows Forms с библиотекой PropertyGrid



Окно планировщика задач

Начало работы программы

Процесс загрузки данных

```
19.05.2025 13:01 hashPassport
19.05.2025 13:00 hashSnils
19.05.2025 13:00 swift
19.05.2025 13:01 fastPayNumber
19.05.2025 13:00 phoneNumber
19.05.2025 13:01 ewalletNumber
19.05.2025 13:00 cardNumber
19.05.2025 13:01 accountNumber
```

Процесс экспорта данных в БД

```
Файл: ewallet_number.csv
-----Импорт в БД успешен-----
-----Начало импорта в БД-----

Обработка таблицы: users_feedsfastpaynumbers
Файл: fastpay_number.csv
-----Импорт в БД успешен-----
-----Начало импорта в БД-----

Обработка таблицы: users_feedsinn
Файл: inn.csv
-----Импорт в БД успешен-----
-----Начало импорта в БД-----

Обработка таблицы: users_feedspassporthash
Файл: passport_hash.csv
-----Импорт в БД успешен-----
-----Начало импорта в БД-----

Обработка таблицы: users_feedsphonenumber
Файл: phone_number.csv
-----Импорт в БД успешен-----
-----Начало импорта в БД-----

Обработка таблицы: users_feedssnilhash
Файл: snils_hash.csv
-----Импорт в БД успешен-----
```

Процесс получение данных от МВД

```
Получение бюллетеней по папкам...
Всего: 3568, в листе: 20.
FinCERT-20250520-01-FEEDS
Получен фид МВД 'feeds_20250520-01.zip'
Пропущен бюллетень FinCERT-20250519-06 - не соответствует текущей дате
Пропущен бюллетень FinCERT-20250519-04 - не соответствует текущей дате
Пропущен бюллетень FinCERT-20250518-04 - не соответствует текущей дате
Пропущен бюллетень FinCERT-20250517-03 - не соответствует текущей дате
Пропущен бюллетень FinCERT-20250516-01-Дайджест - не соответствует текущей дате
```

Работа программы с данными IOC

```
Получение бюллетеней по папкам...
Всего: 3568, в листе: 20.
FinCERT-20250520-04
Получен IOC файл 'FinCERT_20250520-143555.zip'
20250520
C:\IOCDownloads\FinCERT_20250520-143555.zip
Обработано 157 записей
2025-05-20 17:28:50 [Info]: Total blocked IPs: 22203
2025-05-20 17:29:08 [Info]: Total blocked CIDR IPs: 10331002
2025-05-20 17:29:08 [Warning]: Unknown IOC type: 'Type IOC'
2025-05-20 17:29:08 [Warning]: Error resolving hostname 'atomicmet.run': Этот хост неизвестен.
2025-05-20 17:29:09 [Warning]: Error resolving hostname 'datawavej.digital': Этот хост неизвестен.
2025-05-20 17:29:18 [Warning]: Error resolving hostname 'medikosmos.ru': Обычно - это временная ошибка, возникающая
речи разрешения имени узла, и означая, что локальный сервер не получил ответа от полноценного сервера.
2025-05-20 17:29:28 [Warning]: Error resolving hostname 'mednobotik.ru': Обычно - это временная ошибка, возникающая
речи разрешения имени узла, и означая, что локальный сервер не получил ответа от полноценного сервера.
2025-05-20 17:29:39 [Warning]: Error resolving hostname 'ns2.taipower.energy': Этот хост неизвестен.
2025-05-20 17:29:39 [Warning]: Error resolving hostname 'rackilb.digital': Этот хост неизвестен.
2025-05-20 17:29:39 [Warning]: Error resolving hostname 'techchaun.live': Этот хост неизвестен.
2025-05-20 17:29:39 [Warning]: Error resolving hostname 'testcawepn.run': Этот хост неизвестен.
2025-05-20 17:29:49 [Warning]: Error resolving hostname 'zelenotek.ru': Обычно - это временная ошибка, возникающая
речи разрешения имени узла, и означая, что локальный сервер не получил ответа от полноценного сервера.
2025-05-20 17:29:49 [Info]: Total IPs to check: 160
```

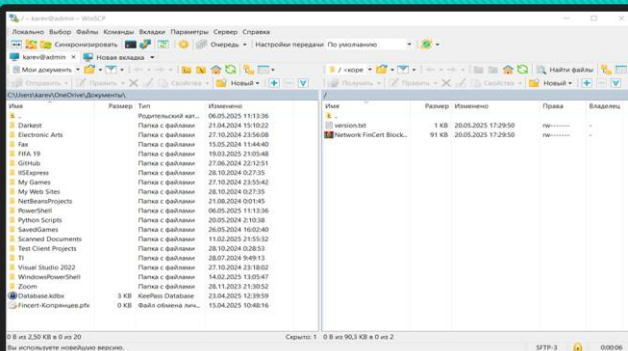
Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.09.001 ПЗ

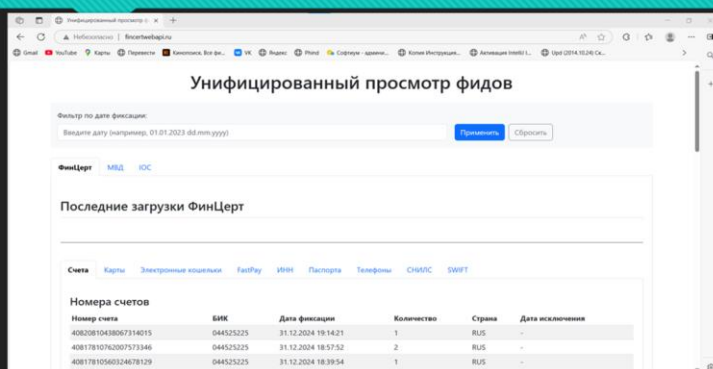
Лист

127

Результат загрузки данных на SFTP-сервер



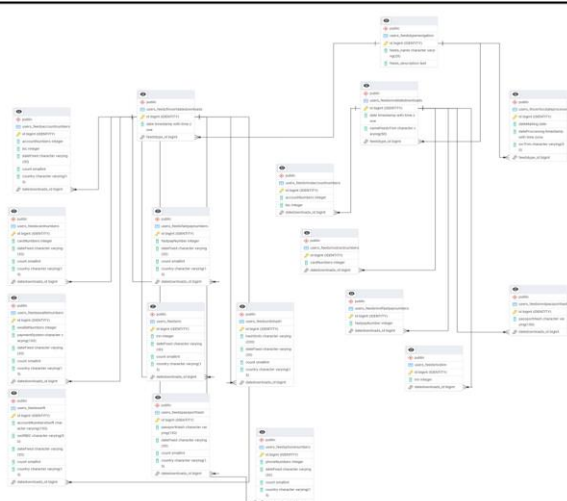
Главная страница веб-приложения



Удобный интерфейс для переключения страниц

40817810051037135607	044525555	31.10.2024 10:43:22	1	RUS	-
40817810710480065858	044525593	31.10.2024 10:37:41	2	RUS	-
40817810151036952448	044525555	31.10.2024 10:28:13	1	RUS	-
40817810751033030747	044525555	31.10.2024 10:09:15	1	RUS	-
40817810351037022290	044525593	31.10.2024 10:02:44	1	RUS	-
40817810506750115985	044525593	31.10.2024 10:02:22	1	RUS	-
40817810205613352898	044525593	31.10.2024 08:42:00	1	RUS	-
4081781015018627881	043469743	31.10.2024 04:27:14	2	RUS	-
40817810510096045734	043601968	31.10.2024 04:19:17	2	RUS	-
40817810963800009829	044525111	31.10.2024 04:15:12	1	RUS	-
40817810890039024148	040813713	31.10.2024 00:32:43	1	RUS	-

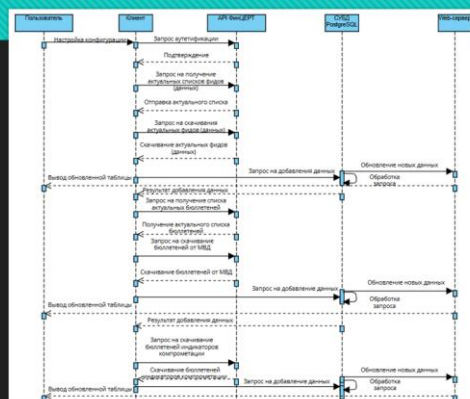
Схема БД



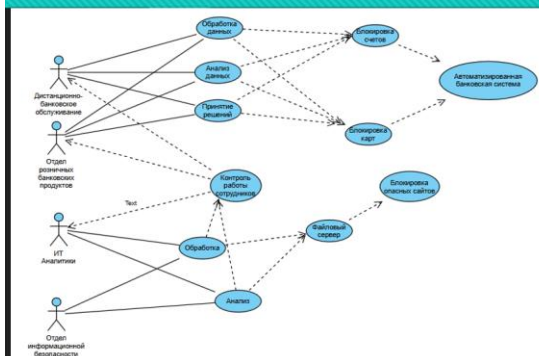
UML-Диаграмма компонентов



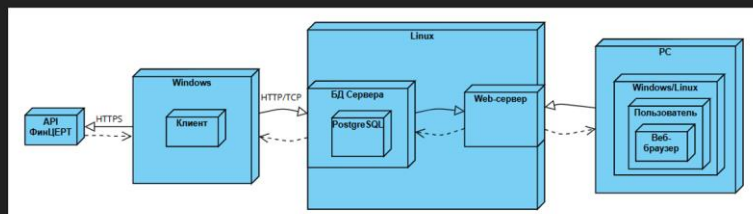
UML-Диаграмма последовательностей



UML-Диаграмма вариантов использования



UML-Диаграмма развертывания



Заключение

Разработанная программа способна выполнять основные требуемые функции.

Также возможно усовершенствовать разработанную автоматизированную систему, добавив:

- добавить аутентификацию и авторизацию для веб-приложения;
- добавить возможность создание отчетов на основе полученных данных;
- Добавить интеграцию с автоматизированной банковской системой.