|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **РОСЖЕЛДОР**  **ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**  **ВЫСШЕГО ОБРАЗОВАНИЯ**  **СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**  **ПУТЕЙ СООБЩЕНИЯ (СГУПС)**   |  |  |  |  | | --- | --- | --- | --- | | **К защите:** |  |  | | | **Заведующий кафедрой** | **Информационные** | | | **технологии транспорта** | | | |  | д-р техн. наук, профессор | | |  |  | В.И. Хабаров | | | *подпись* |  | *инициалы, фамилия* | | |  |  |  | | | *дата* |  |  | |   **ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**  **(БАКАЛАВРСКАЯ РАБОТА)**   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Тема:** | Разработка службы Windows для интеграции с API сервиса отслеживания | | | | | почтовых отправлений Почты России | | | | | |  | | | | | |  |  | БР.БПИ.03.2022 |  |  | |  |  | *шифр документа* |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | |  |  | **Выполнил** |  |  |  | **Руководитель** | |  |  | М.В. Боброва |  |  |  | канд. техн. наук, доц  А.О. Коломеец | | *подпись* |  | *инициалы, фамилия* |  | *подпись* |  | *инициалы, фамилия* | |  |  |  |  |  |  |  | | *дата* |  |  |  | *дата* |  |  |   **Консультанты по разделам**   |  |  |  |  |  | | --- | --- | --- | --- | --- | | 1 |  |  |  |  | |  |  | *подпись* |  | *инициалы, фамилия* | | 2 |  | *дата* |  |  | |  |  | *подпись* |  | *инициалы, фамилия* | |  |  | *дата* |  | ст. преп. | | Нормоконтролер работы |  | *дата* |  | Т.А. Распопина | |  |  | *подпись* |  | *инициалы, фамилия* | |  |  | *дата* |  |   **2022 г.** |

**СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**ПУТЕЙ СООБЩЕНИЯ (СГУПС)**

Факультет: Бизнес-информатики

Кафедра: Информационные технологии транспорта

Направление: 09.03.03 «Прикладная информатика»

Профиль: Корпоративные информационные системы на транспорте

|  |  |
| --- | --- |
|  | ***УТВЕРЖДАЮ****: зав. кафедрой «Информационные технологии транспорта»*  д-р техн. наук, проф.  В. И. Хабаров |
|  | *«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_ 2022 г.* |

**З А Д А Н И Е**

**на выполнение выпускной квалификационной работы**

|  |  |  |  |
| --- | --- | --- | --- |
| студенту | Бобровой Марии Витальевне | | |
|  |  | | |
| 1. Тема «Разработка службы Windows для интеграции с API сервиса отслеживания почтовых отправлений Почты России» утверждена приказом № 203/с от «30» мая 2022 г. | | | |
| 2. Задание выдано «12» мая 2022 г. | | | |
| 3. Срок сдачи законченной работы на кафедру «20» июня 2022 г. | | | |
| 4. Исходные данные: данные, полученные в ходе прохождения преддипломной практики | | | |
| 5. Содержание расчетно-пояснительной записки | | | |
| Наименование разделов и вопросов | | Примерное количество страниц | График (сроки) выполнения |
| Введение | | 2 | 13.05.2022 |
| Аналитическое исследование | | 12 | 13.05.2022 |
| Моделирование бизнес-процессов | | 6 | 15.05.2022 |
| Разработка и апробация службы Windows | | 24 | 28.05.2022 |
| Руководство пользователя | | 8 | 01.06.2022 |
| Заключение | | 1 | 01.06.2022 |

6. Содержание и объемы графической части

|  |  |  |
| --- | --- | --- |
| Наименование графического документа (чертежа, схемы, графика) | Количество  листов  формата А1 | График  (сроки)  выполнения |
| Презентация PowerPoint | 23 | 05.06.2022 |

7. Консультанты по разделам

|  |  |  |
| --- | --- | --- |
| Наименование  раздела | Фамилия, И. О.  консультанта | Подпись консультанта,  дата выдачи задания |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| Руководитель |  | А. О. Коломеец |
|  | *(подпись, фамилия, И.О.)* |  |
| Задание к использованию принял |  | М. В. Боброва |
|  | *(подпись студента)* |  |

УДК 004.41

**АННОТАЦИЯ**

В работе 60 страниц, 66 рисунков, 17 источников.

Ключевые слова*: служба Windows, API, сервис отслеживания, интеграция, программное обеспечение, почтовое отправление, обращения, запросы.*

Проведено исследование, которое выявило назначение служб Windows, их свойства и способы управления. Были разработаны диаграммы в нотации UML для изучения процессов разработки ПО.

В результате была реализована служба Windows для интеграции с API сервиса отслеживания почтовых отправлений Почты России и приложение, которое отображает полученную информацию по отправлению, реализовывает функционал управления службой и настройками периода обращения к сервису отслеживания Почты России.

**ABSTRACT**

The work has 60 pages, 66 drawings, 17 sources.

Keywords: *Windows service, API, tracking service, integration, software, mailing, hits, requests.*

A study was conducted that revealed the purpose of Windows services, their properties and methods of management. Diagrams in UML notation were developed to study software development processes

As a result, a Windows service was implemented to integrate with the API of the Russian Post tracking service and an application that displays the received information on the shipment, implements the functionality of managing the service and settings for the period of access to the Russian Post tracking service.

**ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ**

РПО – регистрируемое почтовое отправление.

ПК – персональный компьютер.

ПО – программное обеспечение.

ПС – программное средство.

ОС – операционная система.

БД – база данных.

СУБД – система управления базами данных.

НФ – нормальная форма.

ПКМ – правая кнопка мыши.

API (Application Programming Interface) – интерфейс прикладного программирования.

IDE (Integrated Development Environment) – интегрированная, единая среда разработки.

SCM (Service Control Manager) – менеджер управления службами.

ER (Entity-Relationship) – сущность-связь.

RTB – RichTextBox.

DGV – DataGridView.

**СОДЕРЖАНИЕ**

[Введение 6](#_Toc105939899)

[1 Аналитическое исследование 8](#_Toc105939900)

[1.1 Общие сведения о службе Windows 8](#_Toc105939901)

[1.2 Способы управления службами Windows 9](#_Toc105939902)

[1.3 Свойства службы Windows 10](#_Toc105939903)

[1.4 API. Определение, виды, архитектурные стили 12](#_Toc105939904)

[1.5 Интеграция API 15](#_Toc105939905)

[1.6 API Сервиса отслеживания Почты России 17](#_Toc105939906)

[2 Моделирование бизнес-процессов 20](#_Toc105939907)

[2.1 Диаграмма вариантов использования 20](#_Toc105939908)

[2.2 Диаграмма последовательности 21](#_Toc105939909)

[2.3 Диаграмма состояний 24](#_Toc105939910)

[2.4 Диаграмма деятельности 25](#_Toc105939911)

[3 Разработка и апробация службы Windows 26](#_Toc105939912)

[3.1 Анализ средств разработки 26](#_Toc105939913)

[3.2 Разработка службы Windows 29](#_Toc105939914)

[3.3 Проектирование базы данных 37](#_Toc105939915)

[3.4 Разработка приложения 40](#_Toc105939916)

[4 Руководство пользователя 50](#_Toc105939917)

[4.1 Общие сведения о Сервисе 50](#_Toc105939918)

[4.2 Назначение и область применения 50](#_Toc105939919)

[4.3 Работа с программным средством 50](#_Toc105939920)

[Заключение 58](#_Toc105939921)

[Список использованных источников 59](#_Toc105939922)

# **ВВЕДЕНИЕ**

В современном мире интерфейсы прикладного программирования (API) являются неотъемлемой частью почти всего в мире цифровых технологий. API применяются во многих отраслях и являются важными инструментами, обеспечивающие высокую производительность и бесперебойную работу веб-систем и приложений.

Чтобы понять значимость интерфейсов прикладного программирования можно подумать о приложениях, которые используются каждый день. API используются для настольных приложений, веб-приложений, мобильных приложений, определяют информацию, передаваемую между приложениями, системами, устройствами и многое другое.

API существуют десятилетиями. Современные веб-API стали стандартным способом обмена данными между системами и приложениями, что делает их жизненной силой онлайн-мира.

Среди программ любой операционной системы и, в том числе — ОС Windows, существуют программы, работающие в фоновом режиме и выполняющие функции посредника. Подобные фоновые процессы называются службами операционной системы.

Служба Windows предназначена для выполнения конкретных операций. Например, существуют службы обновления, служба времени, служба печати, служба безопасности, сетевые службы Windows и другие.

Целью данной работы является разработка службы Windows для интеграции с API сервиса отслеживания почтовых отправлений Почты России.

Объектом исследования является API сервиса отслеживания почтовых отправлений Почты России.

Предметом исследования является интеграция службы Windows с API сервиса почтовых отправлений Почты России.

Для достижения поставленной цели необходимо выполнить следующие задачи:

* изучить понятие и назначение служб Windows;
* изучить понятие интерфейсов прикладного программирования (API);
* ознакомиться c API Сервиса отслеживания почтовых отправлений Почты России;
* произвести моделирование процессов ПО;
* разработать службу Windows для интеграции с API Сервиса отслеживания постовых отправлений Почты России;
* разработать приложение для управления службой и базой данных;
* протестировать разработанное ПО.

Выпускная квалификационная работа состоит из перечня сокращений, введения, четырех разделов, заключения и списка используемых источников.

В первом разделе проведено аналитическое исследование, определяющее понятие службы Windows, основное назначение служб, их свойства и способы управления. Кроме того, раскрыто понятие API, приведена классификация видов и архитектурных стилей. Объяснено понятие интеграции API и описано API сервиса отслеживания почтовых отправлений Почты России.

Во втором разделе проведено моделирование процессов в области разработки программного обеспечения с помощью языка графического описания UML. Разработаны диаграммы четырех видов, а именно диаграммы вариантов использования, последовательности, состояний и диаграммы деятельности.

Третий раздел содержит разработку службы Windows, проектирование базы данных и разработку приложения.

В заключении сделаны выводы по результатам исследования и разработке службы Windows для интеграции с API сервиса отслеживания почтовых отправлений Почты России.

# **Аналитическое исследование**

* 1. **Общие сведения о службе Windows**

Служба Windows – это расширенный компонент Microsoft Windows, которая поддерживает управление длительными фоновыми процессами и приложениями. В отличие от обычных программ, которые запускаются пользователем и запускаются только при входе этого пользователя в систему, приложение службы Windows может запускаться до входа любого пользователя и может продолжать работать даже после выхода всех пользователей. Службы Windows подходят для программного обеспечения, которое должно запускаться при загрузке ПК.

Службы хранятся в [реестре Windows](https://en.wikipedia.org/wiki/Windows_Registry), которая является неотъемлемой частью операционной системы.

Реестр Windows – это [иерархическая база](https://en.wikipedia.org/wiki/Hierarchical_database) данных, в которой хранятся [низкоуровневые](https://en.wikipedia.org/wiki/High-_and_low-level) параметры операционной системы Microsoft Windows и приложений, решивших использовать реестр. [Ядро](https://en.wikipedia.org/wiki/Kernel_(operating_system)), [драйверы устройств](https://en.wikipedia.org/wiki/Device_driver), [службы](https://en.wikipedia.org/wiki/Windows_service), [менеджер учетных записей безопасности](https://en.wikipedia.org/wiki/Security_Accounts_Manager) и [пользовательские интерфейсы](https://en.wikipedia.org/wiki/Graphical_user_interface) могут использовать реестр. Реестр также предоставляет доступ к [счетчикам](https://en.wikipedia.org/wiki/Instrumentation_(computer_programming)) для профилирования производительности системы.

Другими словами, реестр или реестр Windows содержит информацию, настройки, параметры и другие значения для программ и оборудования, установленных во всех версиях операционных систем Microsoft Windows. Например, при установке программы в реестр Windows добавляется новый подраздел, содержащий такие параметры, как расположение программы, ее версия и способ запуска программы. Службы могут использоваться Windows для управления многими вещами, такими как печать, обмен файлами, связь с устройствами Bluetooth, проверка обновлений программного обеспечения, размещение веб-сайта и так далее [1].

Служба может быть установлена даже сторонней программой, отличной от Windows, например, в качестве [инструмента резервного копирования файлов](https://www.lifewire.com/free-backup-software-tools-2617964), [программы шифрования дисков](https://www.lifewire.com/free-full-disk-encryption-programs-2619180), [утилиты онлайн-резервного копирования](https://www.lifewire.com/online-backup-services-reviewed-2624712) и многого другого [2].

Как и любое другое приложение, службы также используют системные ресурсы. При загрузке компьютера сразу происходит поглощение части оперативной памяти службами, даже если запуск приложений не происходит.

Так как службы напрямую влияют на работу компьютера важно знать, как ими управлять.

* 1. **Способы управления службами Windows**

Поскольку службы не открывают и не отображают параметры и окна, для работы с ними необходимо использовать встроенные инструменты Windows.

Один из инструментов является диспетчер управления службами. Другой инструмент управление службами с помощью командной строки, но данный способ более сложен в использовании и поэтому не нужен большинству людей.

Самый простой способ открыть службы – это ярлык службы в [инструментах администрирования](https://www.lifewire.com/administrative-tools-2625804), который доступен через [панель управления](https://www.lifewire.com/control-panel-2625841). Пуск\Панель управления\Система и безопасность Администрирование\Службы.

Для работы через командную строку или диалоговое окно «Выполнить» (запуск происходит нажатием клавиш Win+R) необходимо ввести команду services.msc.

Если работа производится на ОС [Windows 11](https://www.lifewire.com/windows-11-5188930), [Windows 10](https://www.lifewire.com/windows-10-2626217), [Windows 8](https://www.lifewire.com/windows-8-2626235), [Windows 7](https://www.lifewire.com/windows-7-2626265) или Windows Vista, можно просмотреть службы с помощью диспетчера задач.

После выполнения одного из способов управления службами происходит открытие списка установленных служб Windows, рисунок 1.1.

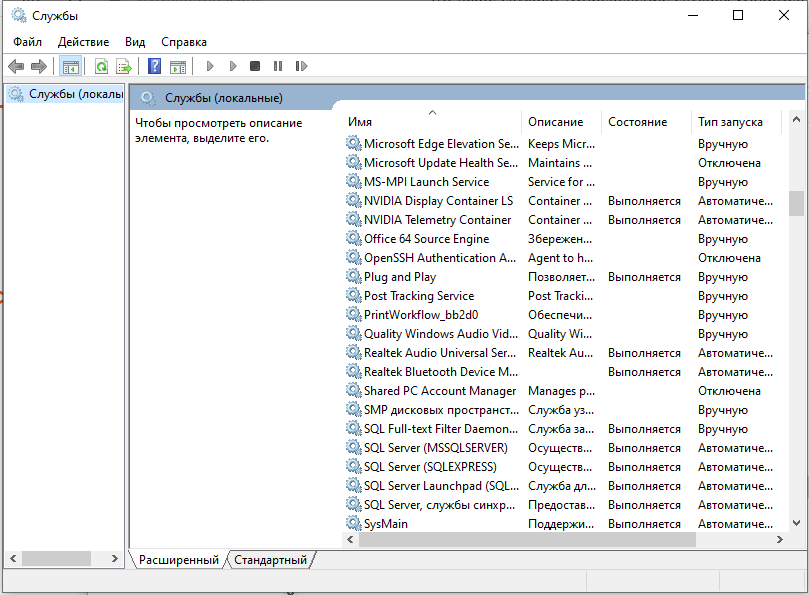


Рисунок 1.1 – Окно «Службы»

Службы можно запускать, останавливать, приостанавливать или возобновлять работу по мере необходимости. Также есть возможность изменить параметры, такие как тип запуска, учетная запись входа в систему и другое.

* 1. **Свойства службы Windows**

Двойной щелчок на любой службе откроет ее конкретные свойства, рисунок 1.2. Имя службы будет показано в верхней части окна свойств. Свойства службы невозможно просмотреть из диспетчера задач, для этого необходимо быть в утилите «Службы».

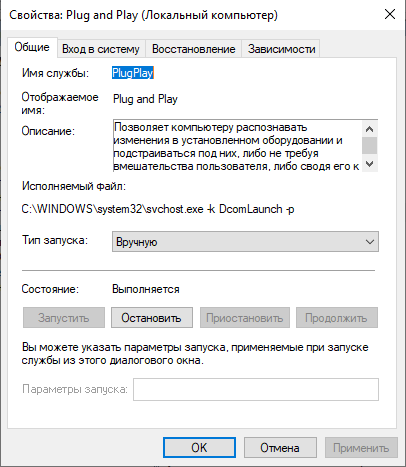


Рисунок 1.2 – Окно «Свойства: «имя\_службы»»

Окно «Свойства» является самой важной частью консоли управления службами Windows, в нем перечислены все элементы управления и информация о службе.

Вкладка «Общие» отображает путь к исполняемому файлу службы и описание того, что она делает. Кроме того, у службы можно выбрать «Тип запуска», чтобы настроить запуск службы. Существуют четыре типа запуска:

* автоматически (отложенный запуск): суть отложенного запуска заключается в том, что служба запускается не сразу при включении компьютера, а после полной его полной загрузки. Если служба не нужна немедленно, то можно выбрать данный тип запуска, это приведет к ускорению загрузки Windows;
* автоматически: при данном типе служба запускается одновременно с запуском Windows;
* вручную: запуск службы происходит при запуске связанного с ней приложения или функции, а также когда работа службы нужна не постоянно и пользователь сам определяет, когда ему необходима работа службы. При таком типе запуска необходимо помнить, что работа служб, которые зависят от основной службы и от мониторинга реального времени, может нарушиться;
* отключено: данный тип произведет отключение службы и сделает ее, это отключит службу и сделает её недоступной для использования.

На вкладке «Зависимости» перечислены все службы, которые зависят от выбранной службы и от которых зависит служба, свойства который просматриваются. Перед тем как менять опцию «Тип запуска» необходимо изучить зависимости служб, чтобы не нарушить работу системы. Так как при отключении службы, зависящие от нее службы, могут либо не работать, либо работать некорректно.

* 1. **API. Определение, виды, архитектурные стили**

API (Application Programming Interface) означает интерфейс прикладного программирования. API – это фрагменты кода, которые позволяют цифровым устройствам, программным приложениям и серверам данных общаться друг с другом, и они являются основой многих современных сервисов. Благодаря этой совокупности инструментов и функций происходит взаимодействие одной программы с другой.

Для лучшего понимания работы API можно сравнить его работу с работой веб-приложений и мобильных приложений. Вторые предназначены для взаимодействия с людьми, в свою очередь API предназначены для использования другими цифровыми системами и приложениями.

Приложения и API делают одно и то же (возвращают данные, контент, изображения, видео и другую информацию), но API не возвращают детали, необходимые для того, чтобы все выглядело понятно и красиво для человека. Происходит получение только необработанных данных и другой машиночитаемой информации.

Существует много разных видов API и много разных способов их категоризации. Наиболее распространенный способ классифицировать API-интерфейсы по характеристике доступа к ним:

* частные API. Эти интерфейсы прикладного программного обеспечения предназначены для улучшения решений и услуг в организации. Внутренние разработчики или подрядчики могут использовать эти API для интеграции ИТ-систем или приложений компании, создания новых систем или приложений для клиентов, использующих существующие системы. Даже если приложения общедоступны, сам интерфейс остается доступным только для тех, кто работает непосредственно с издателем API. Частная стратегия позволяет компании полностью контролировать использование API;
* партнерские API. Партнерские API открыто продвигаются, но делятся ими с деловыми партнерами, подписавшими соглашение с издателем. Общим вариантом использования партнерских API является интеграция программного обеспечения между двумя сторонами. Компания, предоставляющая партнерам доступ к данным или возможностям, выигрывает от дополнительных потоков доходов. В то же время он может отслеживать, как используются открытые цифровые активы, обеспечивать ли сторонние решения, использующие их API, достойный пользовательский опыт и поддерживать корпоративный стиль в своих приложениях;
* публичные API. Также известные как внешние, эти API доступны для любых сторонних разработчиков. Публичная API-программа позволяет повысить узнаваемость бренда и получить дополнительный источник дохода при правильном выполнении. Существует два типа публичных API – открытые (бесплатные) и коммерческие [3].

[Определение открытого API](https://github.com/okffi/open-api-definition/blob/master/en/open-api-definition.md) предполагает, что все функции такого API являются общедоступными и могут использоваться без ограничительных условий. Например, можно создать приложение, которое использует API без явного одобрения со стороны поставщика API или обязательных лицензионных сборов. В определении также говорится, что описание API и любая связанная с ним документация должны быть открыто доступны, и что API можно свободно использовать для создания и тестирования приложений.

Коммерческие пользователи API платят абонентскую плату или используют API на платной основе. Популярный подход среди издателей - предлагать бесплатные пробные версии, чтобы пользователи могли оценить API перед покупкой подписки.

Для быстрой и масштабной интеграции приложений, API реализуются с использованием протоколов и/или спецификаций, определяющих семантику и синтаксис передаваемых сообщений. Эти спецификации составляют архитектуру API. Список наиболее популярных стилей:

* [REST](https://blog.postman.com/rest-api-definition/) (Representational State Transfer) используется для передачи репрезентативного состояния. API REST опирается на несколько руководящих принципов, таких как структура клиент-сервера, простые и универсальные интерфейсы для взаимодействия между системами, операции без состояния и многое другое;
* Webhooks основан на событиях и, проще говоря, представляет собой автоматические сообщения, отправляемые из одной системы в другую в любое время, когда происходит событие. Webhooks также называют «обратными API» как концепцию проверки изменений в данных;
* [SOAP](https://blog.postman.com/soap-api-definition/) (Simple Object Access Protocol) протокол доступа к простым объектам. Является более структурированным, формализованным и надежным, чем другие API, но может быть медленнее. SOAP API использует протокол обмена сообщениями на основе XML, который включает теги конверта <envelope> (тегом начинается и заканчивается каждое сообщение), заголовка (если сообщение должно определять какие-либо особенности или дополнительные условия) и тела содержащего запрос или ответ;
* API WebSocket: полагается на протокол компьютерной связи WebSocket, который представляет собой полнодуплексный канал связи по одному TCP-соединению, предоставляет стандартный способ для серверов отправлять информацию и данные клиентам, даже если клиент не запрашивает данные. API WebSocket также позволяют передавать данные между клиентами и серверами, сохраняя при этом соединения открытыми;
* gRPC API: RPC в gRPC означает удаленный вызов процедуры; API gRPC были созданы Google. В gRPC клиент может вызывать сервер так же, как это локальный объект, что облегчает взаимодействие распределенных приложений и систем друг с другом;
* Server-sent eventAPI также известен как SSE. Технология, опирающаяся на данные, передаваемые с сервера. Это позволяет клиенту автоматически получать обновления через HTTP-соединение;
* AMQP (Advanced Message Queueing Protocol). AMQP – это протокол, который следует открытым стандартам и работает на прикладном уровне. Открытый протокол прикладного уровня для передачи сообщений между компонентами системы. Основная идея состоит в том, что отдельные подсистемы могут обмениваться произвольным образом сообщениями через AMQP-брокер, который осуществляет маршрутизацию, возможно гарантирует доставку, распределение потоков данных, подписку на нужные типы сообщений [4].

С течением времени появились разные архитектурные стили API. Каждый из них содержит собственные схемы стандартизации обмена данными.

* 1. **Интеграция API**

Интеграция API может быть определена как процесс создания средства для двух или более API для обмена данными и связи друг с другом без вмешательства человека. Она включает в [себя использование API](https://rapidapi.com/blog/how-to-use-an-api/) для обеспечения связи между двумя веб-инструментами или приложениями. Это позволяет организациям автоматизировать свои системы, улучшить беспрепятственный обмен данными и интегрировать текущие приложения.

Создаются системы, в которых данные будут передаваться между различными программными средствами без необходимости делать это вручную. Интеграция API оказалась столь необходимым решением, поскольку она позволяет обмениваться данными процессов и предприятий между приложениями. Это повышает гибкость предоставления информации и услуг, а также упрощает встраивание контента с разных сайтов и приложений. API выступает в качестве интерфейса, который позволяет интегрировать два приложения.

В настоящее время существует несколько методов для достижения интеграции API, и они в значительной степени зависят от уникальных потребностей системы или бизнеса:

* пользовательские интеграции. Этот вид интеграции включает в себя использование рукописного кода от разработчика программного обеспечения с опытом в API. Хотя этот метод был известен несколько лет назад, его популярность резко упала из-за появления других простых методов интеграции;
* приложения-соединители. Это приложения, предназначенные для использования API, которые облегчают интеграцию между двумя известными программными платформами. Они массово производятся для использования различными предприятиями;
* платформы управления интеграцией API. Обычно SaaS-приложения, эти платформы предназначены для разработки API-интеграций, которые помогают присоединиться к другим SaaS-приложениям и системам.  
  В отличие от того, когда требуется опытный разработчик, чтобы написать сценарий с нуля, чтобы помочь интегрировать приложения, сегодня возможно создавать собственные приложения интеграции, используя надежные инструменты, доступные на рынке. Некоторые из них, такие как [RapidQL](https://rapidapi.com/blog/rapidql-tutorial-mysql-open-weather-api-twilio-sms-api/), позволяют извлекать и преобразовывать данные в форму, которую приложение может понять с помощью одного запроса. Это позволяет делать запросы к различным API, а также базам данных одновременно [5].

Преимущества интеграции API:

* автоматизация. Интеграция API позволяет автоматически передавать информацию и данные из одного приложения в другое, что раньше делалось вручную сотрудником за заработную плату;
* масштабируемость. Использование интеграции API позволяет предприятиям расти, поскольку им не нужно начинать с нуля при создании подключенных систем и приложений;
* упрощенная видимость, коммуникация и отчетность. Использование интеграции API позволяет вам иметь сквозную видимость систем и процессов для улучшения связи и отчетности. Благодаря оптимизированной системе возможно эффективно отслеживать данные, создавая надежные отчеты, основанные на тщательных и всеобъемлющих наборах данных;
* уменьшение количества ошибок. Интеграция API позволяет передавать сложные и объемные данные с меньшим количеством ошибок и несоответствий. В сравнении с работой человека, использование API интеграции сокращает количество ошибок, особенно когда речь идет о больших и сложных наборах данных.

Интеграция API стала ключевой в современном мире из-за роста объемов использования облачных продуктов и приложений.

* 1. **API Сервиса отслеживания Почты России**

Почта России предусматривает два вида API: API трекинга и API отправки посылок. Каждый из этих видов предназначен для разных целей.

API трекинга используется в случае отправления большого количества бандеролей, посылок или писем и интеграции функции отслеживания отправлений в приложение, сайт или систему. API трекинга позволяет получать такую информацию об отправлении как детальную историю об одном отправлении или целой партии, текущий статус отправления и информацию об отплате наложенных платежей [6].

Второй вид API отправки посылок предназначен для интеграции в приложение функций проверки правильности написания почтовых адресов, автоматический подбор индекса по адресу, вычисление сроков доставки и ее стоимости, получения трек-номера после оформления заказа и другое.

Для написания службы Windows отслеживания почтовых отправлений Почты России используется API трекинга, который в свою очередь предусматривает две спецификации, другими словами два режима доступа: режим Единичного доступа и Пакетного доступа. Основное отличие состоит в количестве отслеживаемых отправлений за один запрос.

За одно обращение к Сервису отслеживания режим Единичного доступа предоставляет возможность получить информацию по одному регистрируемому почтовому оправлению. В полученном ответе можно извлечь следующую информацию:

* перечисление операций, произведенных с почтовым отправлением;
* время и место проведения операции;
* атрибут и код операции;
* название операции и другое.

Режим Единичного и Пакетного доступа осуществлен на основе SOAP протокола.

Режим пакетного доступа предоставляет возможность за один запрос получать информацию сразу о нескольких отправлениях. Один запрос может содержать в себе до 3000 регистрируемых почтовых отправлений. Работа в режиме Пакетного доступа производится в два этапа. Во время первого этапа формируется запрос билета на подготовку информации, который содержит список трек-номеров отправлений, во время второго этапа происходит получение ответа по билету. В билете возвращается информация по каждому отправлению, которая аналогична информации в режиме Единичного доступа.

Однако в отличие от режима Единичного доступа режим Пакетного доступа имеет ряд ограничений:

* количество регистрируемых почтовых отправлений в одном запросе не может быть больше 3000;
* обращение за ответом по билету производится не ранее 15 минут после получения билета;
* повторные обращения по одному и тому же билету не следует производить чаще 1 раза в 15 минут;
* сервис отслеживания хранит ответ по билету 32 часа, после истечения этого периода ответ по билету удаляется [7].

Благодаря API появляется возможность встроить сервисы почты России в приложения, сайты, системы учета и документооборота.

# **Моделирование бизнес-процессов**

Для моделирования бизнес-процессов в области разработки ПО служит язык графического описания UML. Данную систему обозначений можно применять для объектно-ориентированного анализа, а также проектирования [8].

В UML существуют следующие виды диаграмм:

* диаграмма классов;
* диаграмма компонентов;
* диаграмма развертывания;
* диаграмма пакетов;
* диаграмма деятельности;
* диаграмма автомата;
* диаграмма вариантов использования;
* диаграмма коммуникации и последовательности;
* диаграмма обзора взаимодействия;
* диаграмма синхронизации [9].

При моделировании процессов для ПО отслеживания почтовых отправлений Почты России понадобились некоторые из них.

* 1. **Диаграмма вариантов использования**

Диаграмма вариантов использования (use case diagram) – диаграмма, отображающая отношения между акторами и вариантами использования (прецедентами). Актор является калькой с английского Actor, что расшифровывается как действующее лицо.

Проектируемая ИС представляется в форме вариантов использования, с которой производят взаимодействие сущности или акторы. Вариант использования предоставляет описание сервисов, которые доступны актору.

Это значит, что каждый вариант использования определяет набор действий, которая совершает система при диалоге с актором. Диаграмма вариантов использования показана на рисунке 2.1.

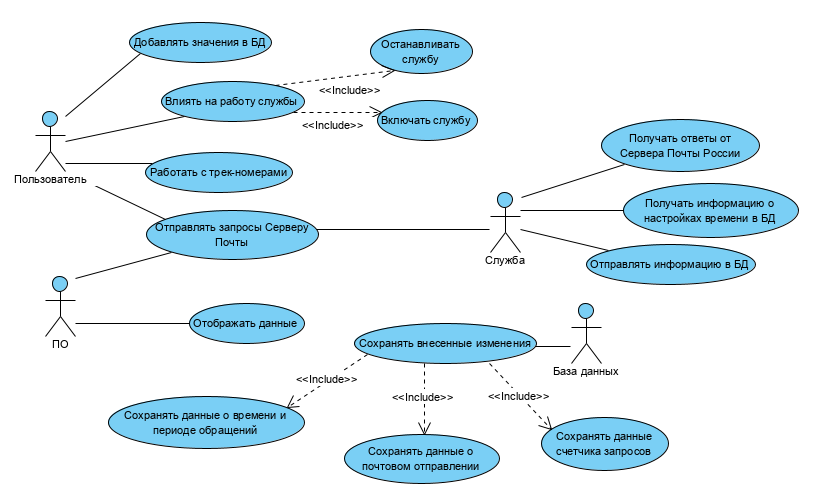


Рисунок 2.1 – Диаграмма вариантов использования

При этом на диаграмме не указывается как будет реализовано взаимодействие системы и актора.

* 1. **Диаграмма последовательности**

Диаграмма последовательности относится к диаграммам взаимодействия UML, описывающим поведенческие аспекты системы, но рассматривает взаимодействие объектов во времени. Другими словами, диаграмма последовательности отображает временные особенности передачи и приема сообщений объектами в рамках одного (конкретного) процесса [10].

На диаграмме последовательности:

* объект (Object) обозначается прямоугольником, в котором указывается информация об участнике действий (название объекта и его класс);
* линия жизни (Life Line) обозначается пунктирной линией, идущей вниз от участника, обозначающая отведенное объекту время жизни;
* активация, фрагмент выполнения (Activation Bar) обозначается узким прямоугольником, расположенным на линии жизни, и указывает начало и завершение действия, в котором участвует объект;
* сообщение (Message) - линия со стрелкой; возвращаемый результат - пунктирная линия со стрелкой.

В диаграмме последовательности существуют дополнительные обозначения. И для циклов, и для условий используются фреймы взаимодействий (interaction frames), представляющие собой средство разметки диаграммы взаимодействия. На рисунке 2.2 показана диаграмма последовательности для автоматического расчета времени. На рисунке 2.3 показана диаграмма последовательности для расчета обращений с учетом рабочего времени. На рисунке 2.4 изображена диаграмма последовательности для добавления трек-номеров.

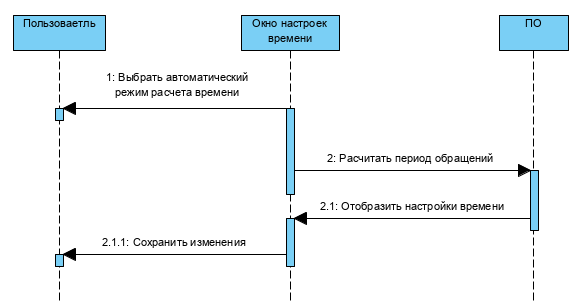


Рисунок 2.2 – Автоматический расчет периода обращений

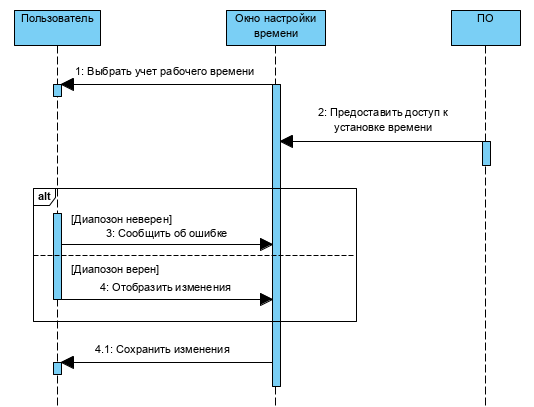


Рисунок 2.3 – Учет рабочего времени

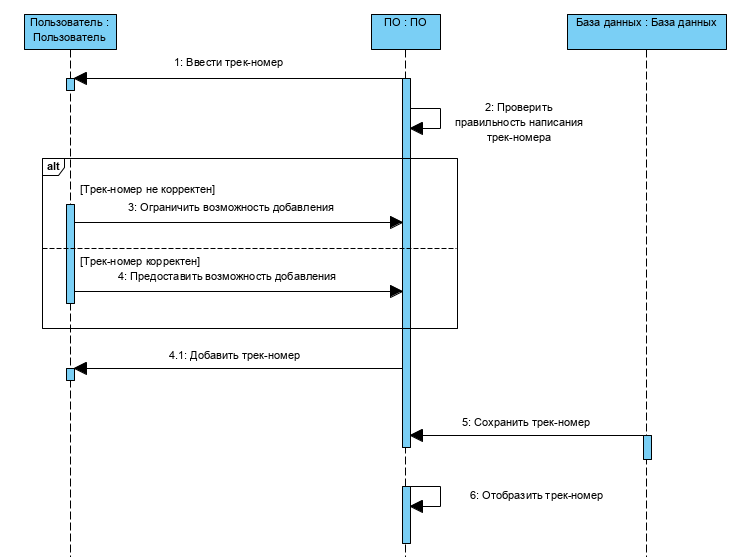


Рисунок 2.4 – Добавление трек-номера

Для диаграммы последовательности ключевым моментом является именно динамика взаимодействия объектов во времени.

* 1. **Диаграмма состояний**

Диаграмма состояний (state diagram) определяет все возможные состояния, в которых может находиться конкретный объект, а также процесс смены состояний объекта в результате влияния некоторых событий. Диаграммы состояний строятся для единственного класса и описывают поведение единственного объекта. Диаграмма состояний представляет собой граф состояний, в которых может находиться объект и связей между ними [11].

Данная диаграмма отображает объект, а также в ней идёт описание пассивного элемента в модели. Диаграмма состояний для установки настроек времени пользователем представлена на рисунке 2.5.

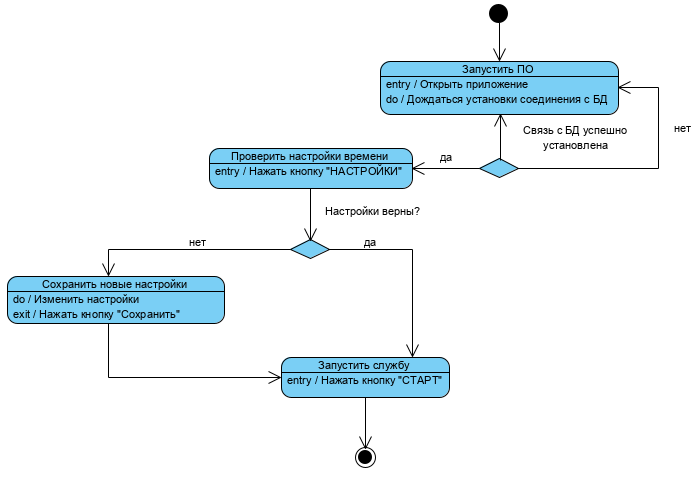


Рисунок 2.5 – Установка времени

Преимуществом является показ возможных состояний, в которых может находиться объект, а также процесс смены состояний в результате внешнего влияния.

* 1. **Диаграмма деятельности**

Диаграммы деятельности или диаграммы видов деятельности, были введены в язык UML сравнительно недавно. Диаграмма деятельности отображает внутрисистемную точку зрения на прецедент.

Диаграммы деятельности применяют для описания шагов, которые должна предпринять система после того, как инициирован прецедент. На рисунке 2.6 показана диаграмма деятельности, которая отображает работу службы и установку времени обращений.

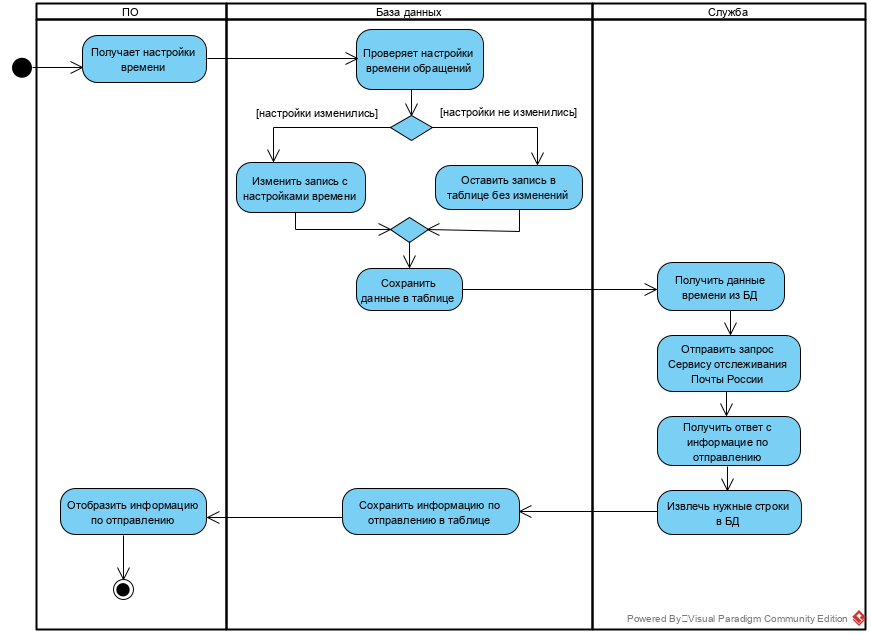
****

Рисунок 2.6 – Работа службы после установки времени

Диаграмма активностей - это, по существу, блок-схема, которая показывает, как поток управления переходит от одной активности (деятельности) к другой, при этом внимание фиксируется на результате деятельности.

# **Разработка и апробация службы Windows**

* 1. **Анализ средств разработки**

Программный продукт разрабатывается для операционной системы Windows. Для написания службы Windows и приложения, с которым осуществляет работу пользователь, используется язык программирования C# и среда разработки Microsoft Visual Studio 2019.

C# является современным объектно-ориентированным языком программирования, который дает возможность разработчикам создавать множество разных типов безопасных и надежных программных компонентов. Являясь частью семейства языков С, наследует ряд функций удобных и полезных в процессе разработки [12]. Некоторые функции, которые помогают в создании устойчивых приложений:

* обработка исключений делает более простым процесс поиска ошибок и помогает восстановить работу программы после их обнаружения;
* все типы языка C#, в том числе и типы-примитивы наследуются от одного класса object, что приводит к наличию единой системы типов;
* сборка мусора приводит к оптимизации использования памяти посредством освобождения ее от неиспользуемых объектов;
* возможность поддержки асинхронных операций;
* поддержка ссылочных типов, определяемых пользователем, и типов значений;
* динамическое выделение объектов и хранение упрощенной структуры в стеке;
* типы, которые допускают значение null и другое.

Кроме того, C# поддерживает основные принципы объектно-ориентированного программирования такие, как абстракция, инкапсуляция, наследование и полиморфизм.

Одной из платформ, которая поддерживает написание программного обеспечения на языке программирования C# является Microsoft Visual Studio. Линейка продуктов Visual Studio включает в себя интегрированную среду разработки программных компонентов, которые включают в себя приложения с графическим интерфейсом, веб-сайты, веб-приложения, а также службы и другие инструменты.

Visual Studio принято считать лучшей IDE для C# из-за того, что оба этих продукта принадлежат компании Microsoft и как следствие подходят для работы друг с другом. Для разработчика Visual Studio удобна в процессе написания программ, так как обладает интуитивно понятным интерфейсом и удобной навигацией. Также Microsoft Visual Studio имеет ряд преимуществ:

* кроссплатформенность разработки;
* поддержка большого количества языков программирования;
* контроль за выполнением многопоточного кода и так далее.

Кроме того, Microsoft Visual Studio 2019 обладает множеством удобных функций, которые способствуют повышению производительности работы программиста:

* рефакторинг кода помогает упростить его понимание, обслуживание и расширение, включая возможности перемещение элементов в интерфейс, настройку пространства имен согласно структуре папок, преобразование циклов и так далее;
* обеспечение контекстно зависимого завершение кода;
* выполнение поиска ошибок в коде, которые трудно выявить;
* при работе в команде помогает придерживаться одного стиля написания кода;
* усовершенствованная система точек останова, которая позволяет расставить их не только в коде, но и в данных;
* внепроцессная обработка символов позволяет выполнять отладку программы, не вызывая проблемы нехватки памяти;
* упрощенный поиск объектов по их свойствам в процессе отладки благодаря инструменту «Закрепляемые свойства»;
* система для скачивания и обновления дополнительных функций;
* встроенные тесты производительности и улучшенная работа с крупными наборами тестов;
* возможность работать с эмулятором мобильных платформ при разработке мобильных приложений [13].

Все перечисленные преимущества и функции указывают на то, что Microsoft Visual Studio 2019 является подходящей платформой для разработки программных продуктов.

В разрабатываемом программном обеспечении производится работа с базой данных, которая хранит в себе трек-номера отправлений и информацию по каждому отправлению. Для взаимодействия прикладного программного обеспечения с базами данных был выбран декларативный язык программирования SQL, который является одним из самых распространённых лингвистических средств работы с БД.

SQL применяется для «создания, модификации и управления данными в реляционной базе данных, управляемой соответствующей системой управления базами данных». SQL справляется с основными требованиями работы с базами данных, а именно их сохранение, изменение, поддержание структуры, масштабирование и разграничение доступа.

К преимуществам языка SQL можно отнести:

* стандартизация использования языка международными организациями;
* возможность переноса на другую СУБД и вычислительную систему;
* понятная табличная структура языка, основа которого является реляционной;
* поддержка клиент-серверной архитектуры;
* возможность программного доступа к БД.

В качестве системы управления базами данных была выбрана Microsoft SQL Server Management Studio 2018. Преимуществами данного решения являются тесная интеграция с операционной системой Windows, высокая отказоустойчивость, высокая производительность.

Основная причина выбора Microsoft SQL Server Management Studio является интеграция с продуктами Microsoft, а именно с Microsoft Visual Studio 2019. Данная СУБД позволяет повысить эффективность применения хранилищ данных и упростить работу с ними.

* 1. **Разработка службы Windows**

Разработка службы Windows начинается с создания нового проекта в Microsoft Visual Studio, который будет иметь тип Служба Windows. Имя решения «PostTrackingService», рисунок 3.1.

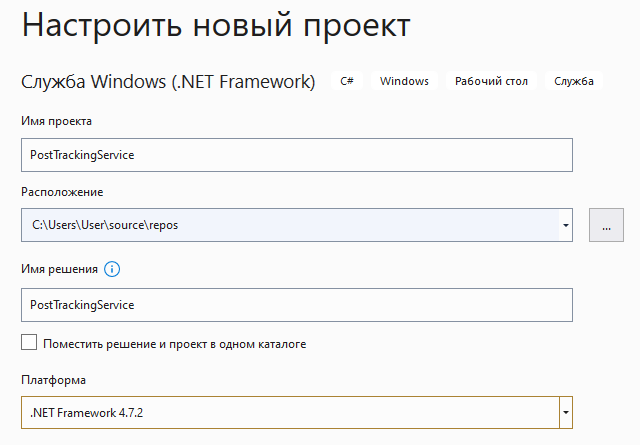


Рисунок 3.1 – Создание проекта

Далее Visual Studio генерирует проект службы как показано на рисунке 3.2. В проекте присутствует файл Program.cs и непосредственно узел службы PostTracking.cs.

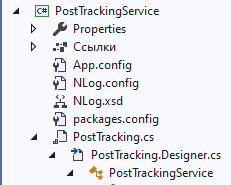


Рисунок 3.2 – Проект «PostTrackingService»

Служба является обычным приложением, но которое не запускается самостоятельно. Обращения и вызовы к службе происходят через SCM или менеджер управления службами.

В процессе разработки службы Windows важно, чтобы была возможность отлаживать ее так же, как и обычное настольное приложение. Отладка служб Windows не является простой, но есть возможность настроить проект для поддержки отладки.

При запуске службы автоматически или вручную, то SCM производит обращение к методу Main в классе Program. Основное отличие от обычного приложения состоит в том, что первое, что происходит в методе Main – это регистрация службы или служб, которые запускаются в контексте диспетчера управления службами, как показано на рисунке 3.3.

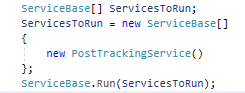


Рисунок 3.3 – Метод Main класса

В методе Main происходит запуск служб, которые определены в массиве ServiceToRun, а запуск производится с помощью метода Run (ServiceBase.Run(ServicesToRun)). Однако после вызова метода Run, появляется диалоговое окно, показанное на рисунке 3.4, которое сообщает о том, что служба не может быть запущена, если она не установлена.

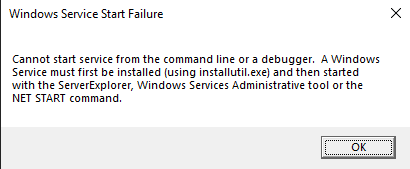


Рисунок 3.4 – Диалоговое окно запуска службы Windows

Чтобы запустить службу, в первую очередь ее необходимо установить с помощью утилиты установки службы, но обычно этого не требуется пока идет процесс разработки. Поэтому метод Main будет изменен путем добавления поддержки отладки с помощью подхода, который называется «Debug service» [14]. На рисунке 3.5 показано как метод Main будет выглядеть с применением этого подхода.

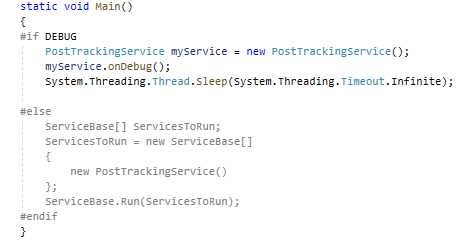


Рисунок 3.5 – Измененный метод Main

Приведенный выше код сначала создает новый экземпляр объекта PostTrackingService, в месте, где определена константа DEBUG, которая будет запускать код при конфигурации решения Debug. После этого вызывается метод onDebug(), который будет вызывать метод OnStart() в классе службы, как показано на рисунке 3.6.

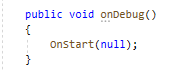


Рисунок 3.6 – Метод onDebug()

Далее происходит вызов метода Sleep(), который ожидает бесконечное количество времени, позволяя служебному коду продолжать выполнение до тех пор, пока отладка не будет остановлена.

Сама служба в программе представлена узлом PostTracking.cs, однако это не просто файл с кодом, в нем содержатся файл дизайнера службы PostTracking.Designer.cs и класс PostTrackingService, рисунок 3.7. Именно класс PostTrackingService представляет службу.



Рисунок 3.7 – Узел PostTracking.cs

Класс службы наследуется от базового класса ServiceBace, данный класс определяет методы, которые важны для работы службы. Одни из самых важных являются методы OnStart() и OnStop(). Первый метод производит запуск действий, выполняемых службой. Второй метод производит остановку службы.

В конструкторе класса PostTrackingService вызывается метод InitializeComponent(), который определен в файле дизайнера PostTracking.Designer.cs, в нем необходимо установить название службы, которое будет отображаться в консоли служб после ее установки, рисунок 3.8.

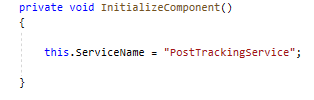


Рисунок 3.8 – Метод InitializeComponent()

Далее изменим код службы и допишем все необходимые методы, рисунок 3.9. Основным классом, который выполняет инкапсуляцию всей функциональности, является класс Logger. С помощью метода Watcher происходит мониторинг изменений, внесенных пользователем в приложение, метод ReceiveAndUpdateData() возвращает историю операций, совершенных над почтовым отправлением рисунок 3.10. Метод TimeCalculation(), вычисляет время отправления запроса Сервису отслеживания Почты России в соответствии с параметрами указанными пользователем.

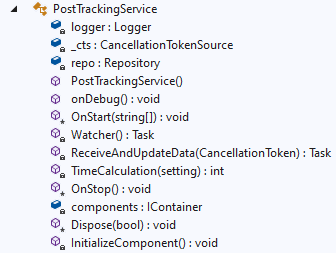


Рисунок 3.9 – Методы класса PostTrackingservice



Рисунок 3.10 – Получение истории операций

Следующим этапом происходит создание установщика службы. Для этого понадобится класс установщика, рисунок 3.11.

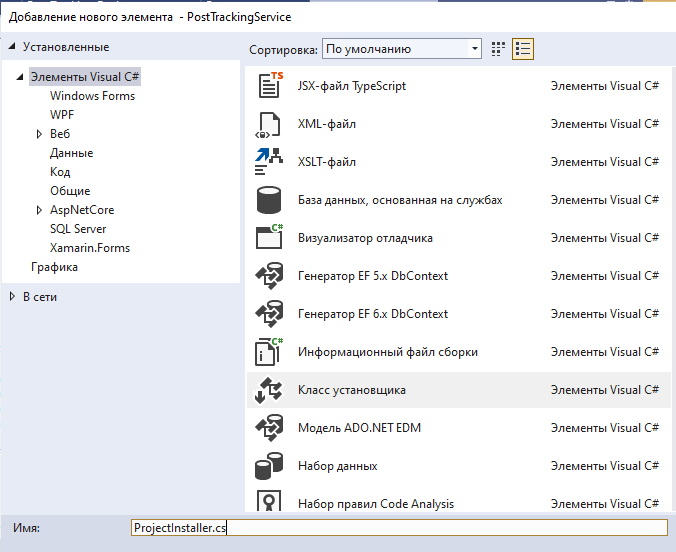


Рисунок 3.11 – Создание класса установщика

После добавления класса установщика узел установщика содержит два файла, а именно файл класса ProjectInstaller и файл дизайнера ProjectInstaller.Designer.cs, рисунок 3.12. Содержание файла кода показано на рисунке 3.13.



Рисунок 3.12 – Узел установщика

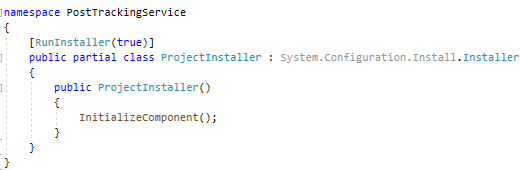


Рисунок 3.13 – Код файла ProjectInstaller.cs

Класс установщика наследуется от класса System.Configuration.Install.Installer, атрибут [RunInstaller(true)] говорит о том, что вызов класса ProjectInstaller должен происходить при установке службы [15].

Для того, чтобы произошла установка службы понадобится два класса ServiceInstaller, который производит настройку для каждой из служб, если их несколько, и ServiceProcessInstaller, который управляет настройкой значений для всех служб. Из-за того, что запускаемая служба одна, объекты двух классов будут в одном экземпляре.

Основные свойства ServiceProcessInstaller:

* Username: задает, под какой учетной записью происходит запуск службы;
* Password: определяет пароль для запуска службы;
* Account: позволяет определить тип учетной записи. Может принимать одно из значений перечисления ServiceAccount: LocalSystem, LocalService, NetworkService и User.

Основные свойства ServiceInstaller:

* DisplayName: устанавливает имя службы, которое будет отображаться в различных утилитах управления службами;
* ServiceName: имя службы, должно совпадать со значением свойства ServiceName у класса службы.

Значения данных перечисленных свойств показаны на рисунке 3.14.

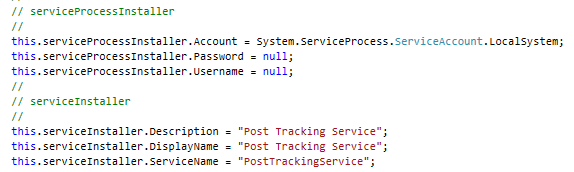


Рисунок 3.14 – Свойства ServiceProcessInstaller и ServiceInstaller

В конце оба объекта установщиков необходимо добавить в коллекцию Installers как показано на рисунке 3.15.



Рисунок 3.15 – Добавление объектов в коллекцию

После того как код службы и код инсталлятора готовы, необходимо скомпилировать проект. Появится исполняемый файл службы, который будет иметь расширение exe. Для установки запускается утилита InstallUtil.exe и ей передается имя файла службы PostTrackingService.exe.

В командной строке происходит открытие утилиты с помощью команды «cd» и указания пути «C:\Windows\Microsoft.NET\Framework\v4.0.30319».

Передача исполняемого файла происходит с помощью команды «installutil» и указания пути исполняемого файла «C:\Users\ User\ Desktop\ Почта\ RussianPostTracking-20220411\ RussianPostTracking\ RussianPostService\ PostTrackingService.exe».

В результате установленную службу можно найти в консоли служб, рисунок 3.16.

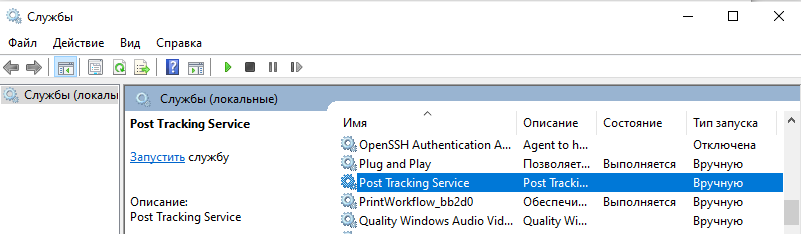


Рисунок 3.16 – Установленная служба

Установка службы Post Tracking Service произошла успешно.

* 1. **Проектирование базы данных**

Для проектирования базы данных необходимо построить концептуальную модель, которая представляется в виде диаграммы сущность-связь, то есть ER-диаграммы. Диаграмма представляется в виде блок-схемы, в которой показано как разные «сущности» связаны друг с другом внутри системы.

Чаще всего используются при проектировании и отладки реляционных баз данных в сфере образования, исследования и разработки программного обеспечения и информационных систем для бизнеса.

Данный процесс называется ER-моделированием. Разработанная ER-диаграмма представлена на рисунке 3.17.

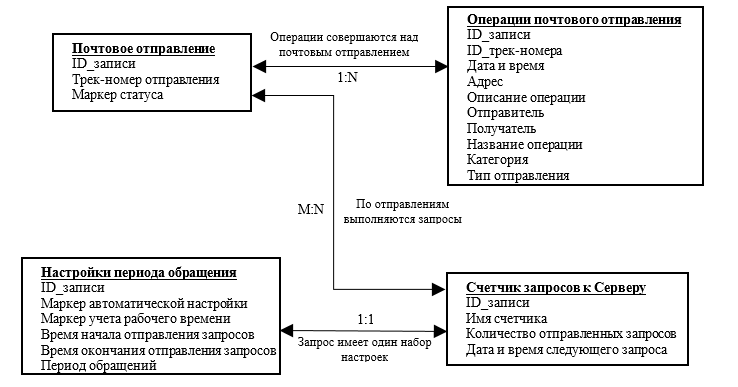


Рисунок 3.17 – ER-диаграмма

Далее концептуальная модель представляется в терминах реляционной модели. Для этого используется формальный аппарат, который позволяет оптимизировать и нормализовать схемы отношений. Первичные ключи обозначены красным, внешние ключи обозначены зеленым, в каждом столбце указаны допустимые значения. На рисунках 3.18-3.21 показаны концептуальные модели в терминах реляционной модели.

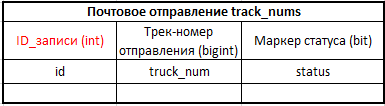


Рисунок 3.18 – Таблица «Почтовое отправление»

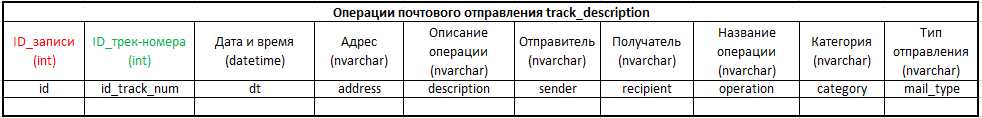


Рисунок 3.19 – Таблица «Операции почтового отправления»

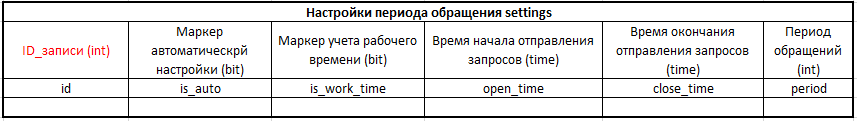


Рисунок 3.20 – Таблица «Настройки периода обращения»

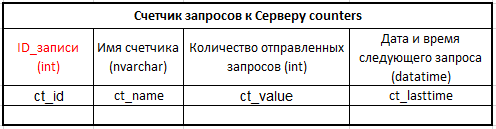


Рисунок 3.21 – Таблица «Счетчик запросов к Серверу»

Схемы отношений должны бать нормализированы по трем формам.

Схемы соответствуют первой нормальной форме (1НФ), если каждая запись в БД представляет один экземпляр сущности. Кроме того, каждая таблица имеет первичный ключ, желательно в виде числового инкрементного поля. В созданных таблицах все первичные ключи представлены числом и поля не имеют дубликатов.

Вторая нормальная форма (2НФ) говорит о том, что поля с не первичным ключом не должны быть зависимы от первичного ключа, а также данные, хранимые в таблице, должны быть напрямую связаны с ней и не относиться к другой сущности.

В третьей нормальной форме (3НФ) все не ключевые атрибуты отношения должны быть взаимно независимы, но должны зависеть от первичного ключа.

Созданная база данных post\_tracking показана на рисунке 3.22.

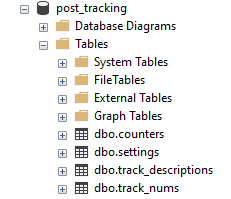


Рисунок 3.22 – База данных post\_tracking

Все созданные таблицы соответствуют правилам нормализации.

* 1. **Разработка приложения**

Приложение создается для взаимодействия пользователя с базой данных и службой Windows. Приложение должно поддерживать следующие функции:

* добавление трек-номеров почтовых отправлений в БД;
* поиск почтовых отправлений по трек-номеру;
* остановка и возобновление отправлений запросов по определенному почтовому отправлению Сервису отслеживания Почты России;
* просмотр информации по почтовому отправлению;
* остановка и возобновление работы службы Windows;
* настройка времени и периода отправлений запросов Сервису отслеживания Почты России.

Приложение для отслеживания почтовых отправлений подразумевает два окна. Первое окно «Трекинг почтовых отправлений» будет отображать базу данных с трек-номерами почтовых отправлений, информацию с операциями по каждому отправлению, поле ввода трек-номера, информацию о запросах к серверу почты, кнопки запуска/остановки службы Windows и кнопку вызова второго окна.

Второе окно «Настройка времени опроса» будет отображать количество активных трек-номеров, интервал опроса, периоды начала работы и окончания работы, а также возможность автоматической настройки времени опроса.

Для реализации описанного функционала понадобятся следующие элементы управления:

* [DataGridView](https://docs.microsoft.com/ru-RU/dotnet/api/system.windows.forms.datagridview) представляет собой настраиваемую таблицу, которая отображает данные;
* [TextBox](https://docs.microsoft.com/ru-RU/dotnet/api/system.windows.forms.textbox) служит для отображения текста, который вводит пользователь;
* [RichTextBox](https://docs.microsoft.com/ru-RU/dotnet/api/system.windows.forms.richtextbox) отображает текст с форматированием в текстовом формате или в формате RTF;
* Label необходим для отображения текста, который недоступен для изменения;
* [ComboBox](https://docs.microsoft.com/ru-RU/dotnet/api/system.windows.forms.combobox) производит отображение раскрывающегося списка элементов;
* [CheckBox](https://docs.microsoft.com/ru-RU/dotnet/api/system.windows.forms.checkbox) отображает флажок рядом с текстом;
* [DateTimePicker](https://docs.microsoft.com/ru-RU/dotnet/api/system.windows.forms.datetimepicker) позволяет пользователю выбрать дату или время;
* [MenuStrip](https://docs.microsoft.com/ru-RU/dotnet/api/system.windows.forms.menustrip) служит для создания контекстного меню;
* [Button](https://docs.microsoft.com/ru-RU/dotnet/api/system.windows.forms.button) является кнопкой и необходим для управления процессами;
* [Panel](https://docs.microsoft.com/ru-RU/dotnet/api/system.windows.forms.panel) необходим для группировки элементов управления [16].

Интерфейс первого окна «Трекинг почтовых отправлений» представлен на рисунке 3.23.

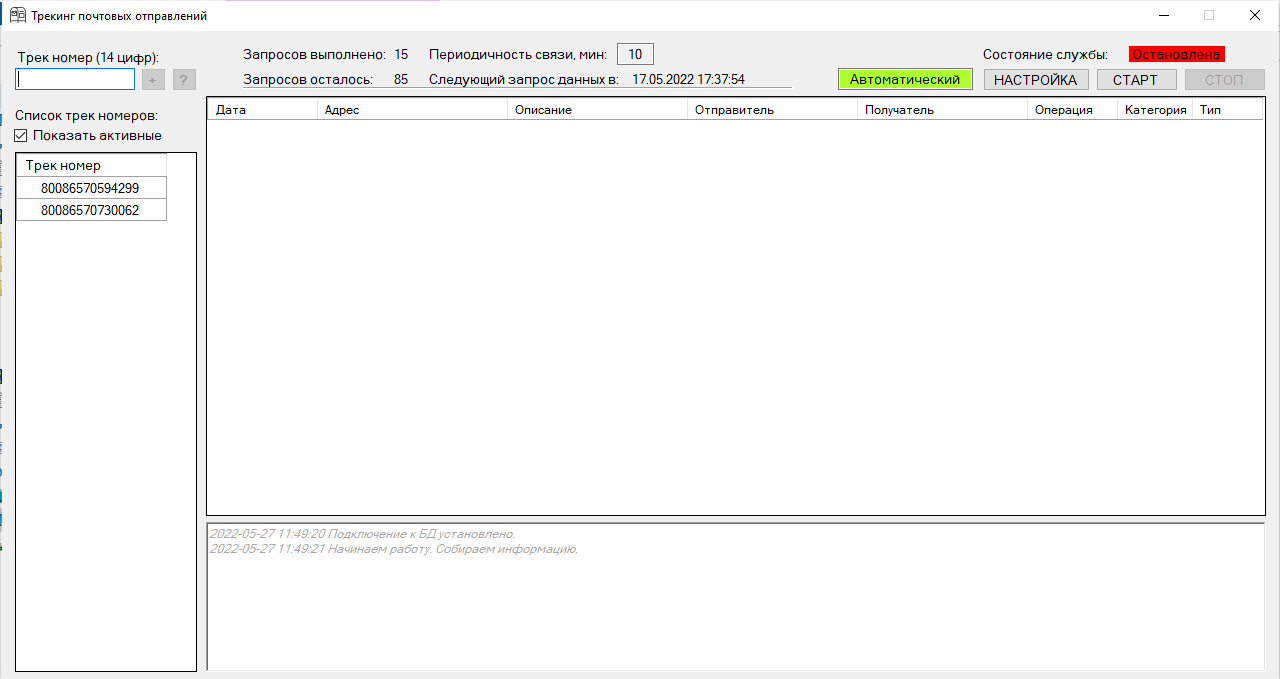


Рисунок 3.23 – Окно «Трекинг почтовых отправлений»

Для добавления трек-номера в базу данных служит событие Btn\_add\_track\_num\_Click, код события показан на рисунке 3.24.



Рисунок 3.24 – Добавление трек-номера в БД

Поиск трек-номера в базе данных осуществляет событие Btn\_find\_tn\_Click, показанное на рисунке 3.25.



Рисунок 3.25 – Поиск трек-номера в БД

Для отображения трек-номеров служит элемент управления DataGridView с именем dgv\_track\_nums. Для каждого трек-номера можно вызвать контекстное меню, которое поможет вызвать информацию по отправлению из базы данных или отправить запрос к Сервису отслеживания Почты России, а также включить или выключить отслеживание трек-номера. Создание контекстного меню представлено на рисунке 3.26.

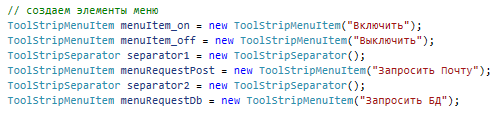


Рисунок 3.26 – Создание элементов контекстного меню

Получение информации об отправлении из базы данных происходит с помощью метода MenuRequestDb\_Click, рисунок 3.27.

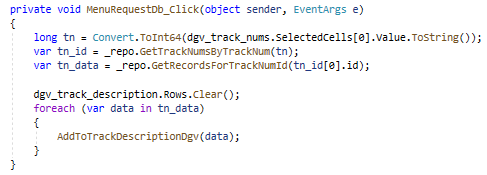


Рисунок 3.27 – Получение информации из БД

Остановка отслеживания по почтовому отправлению происходит с помощью метода MenuItem\_off\_Click. Код метода показан на рисунке 3.28.

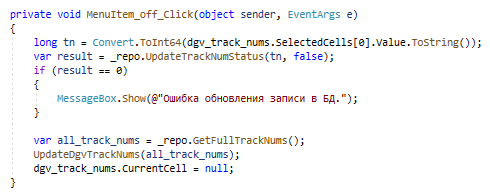


Рисунок 3.28 – Остановка отслеживания почтового отправления

Для возобновления отслеживания операций, совершенных с почтовым отправлением, служит метод MenuItem\_on\_Click, рисунок 3.29.

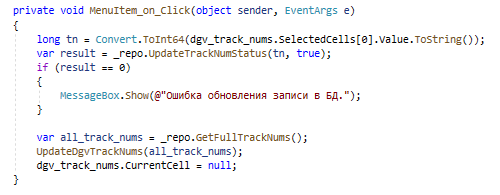


Рисунок 3.29 – Возобновление отслеживания почтового отправления

Для отображения операций, совершенных с почтовым отправлением, служит элемент управления DataGridView с именем dgv\_track\_description. Добавление информации с описание операций происходит с помощью метода AddToTrackDescriptionDgv, рисунок 3.30.

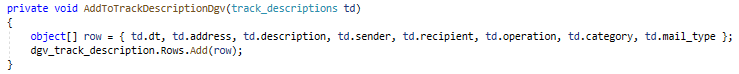


Рисунок 3.30 – Добавление операций в DataGridView

При нажатии на кнопку «СТАРТ» происходит запуск службы PostTrackindService с помощью вызова события Btn\_start\_Click. Код события показан на рисунке 3.31.



Рисунок 3.31 – Старт службы PostTrackingService

После нажатия на кнопку «СТОП» происходит вызов метода Btn\_stop\_Click, который останавливает работу службы. Код метода представлен на рисунке 3.32.



Рисунок 3.32 – Остановка службы PostTrackingService

Для обновления элементов управления Label и TextBox с информацией по количеству отправленный запросов, режиму настройки времени и состоянию работы службы был написан метод UpdateLabels, рисунок 3.33.

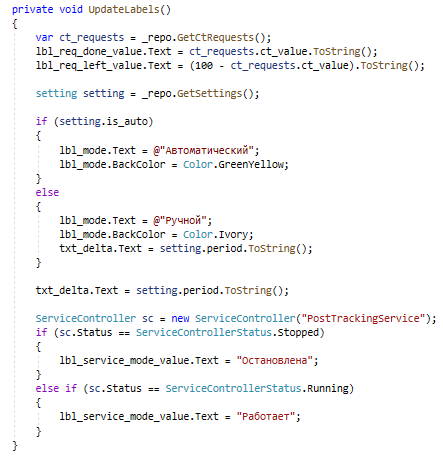


Рисунок 3.33 – Обновление элементов управления Label и TextBox

Нажатие на кнопку «НАСТРОЙКА» вызывает новое диалоговое окно frmCalculation. Для этого был написан метод Bt\_settings\_Click, показанный на рисунке 3.34.

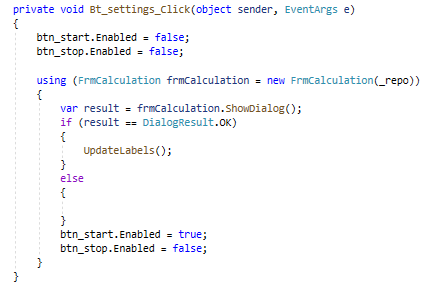


Рисунок 3.34 – Вызов окна frmCalculation

Интерфейс окна «Настройки времени опроса» представлен на рисунке 3.35.

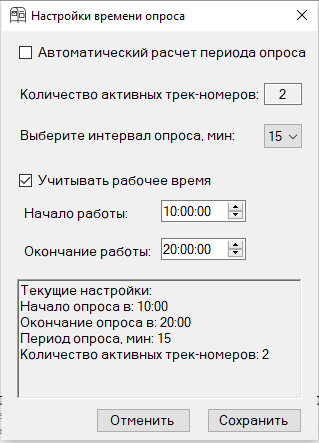


Рисунок 3.35 – Окно «Настройки времени опроса»

Установка флажка возле надписи: «Автоматический период расчета опросов» скрывает возможность выбирать интервал опроса и изменяет настройки времени опроса. Это происходит с помощью события Chkb\_is\_auto\_CheckedChanged, рисунок 3.36.

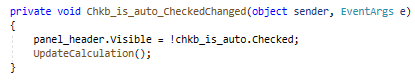


Рисунок 3.36 – Событие Chkb\_is\_auto\_CheckedChanged

Установка флажка возле надписи: «Учитывать рабочее время» делает доступным возможность установки времени начала и окончания работы. Если флажок установлен не будет, то данная панель скрывается. Событие Chkb\_worktime\_CheckedChanged показано на рисунке 3.37.

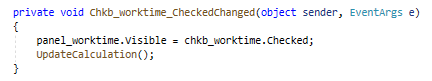


Рисунок 3.37 – Событие Chkb\_worktime\_CheckedChanged

В обоих событиях происходит вызов метода UpdateCalculation, который производит перерасчет настроек отправления запросов, рисунок 3.38, и обновляет данные в элементе управления RichTextBox, рисунок 3.39.

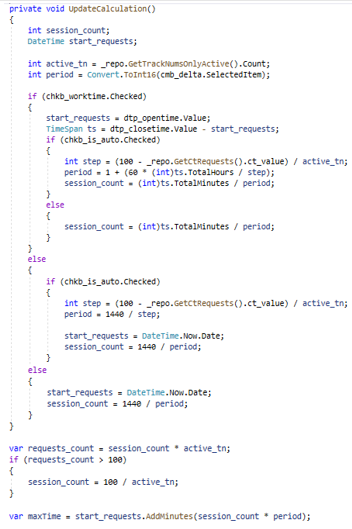


Рисунок 3.38 – Расчет настроек времени отправления запросов

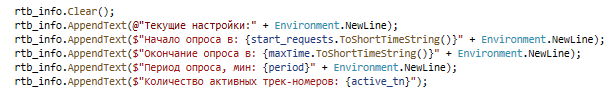


Рисунок 3.39 – Обновление данных в RTB

Нажатие на кнопку «Отмена» производит отмену всех изменений и закрывает окно «Настройки времени». Данный функционал реализуется в событии Btn\_cancel\_Click , рисунок 3.40.

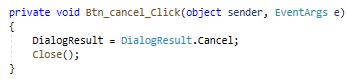


Рисунок 3.40 – Событие Btn\_cancel\_Click

При нажатии на кнопку «Сохранить» происходит проверка значений, заданных пользователем, присваивание переменным новых значений и закрытие окна, в случае корректных значений. Код события Btn\_save\_Click представлен на рисунке 3.41.

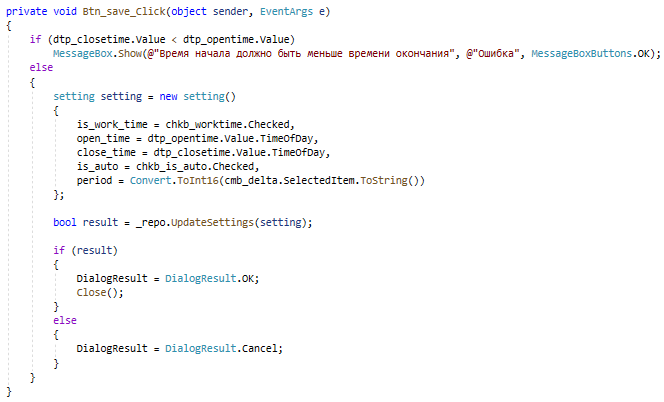


Рисунок 3.41 – Событие Btn\_save\_Click

Подробная инструкция к программе представлена в разделе «Руководство пользователя».

# **Руководство пользователя**

* 1. **Общие сведения о Сервисе**

Сервис отслеживания статуса почтовых отправлений (Post Tracking Service) с использованием API сервиса отслеживания почтовых отправлений Почты России (далее Сервис) реализован в виде службы Windows. При реализации использовался .Net Framework 4.7.2, в качестве СУДБ – MS SQL Server 2018 (без использования функций недоступных для более ранних версий). Сервис работает с базой данных «post\_tracking».

* 1. **Назначение и область применения**

Сервис работает с базой данных имеющихся трек-номеров и осуществляет запросы к API сервиса отслеживания Почты России. API сервиса Почты России работает по протоколу SOAP 1.2 (единичный режим).

Сервис имеет возможность отправлять запросы в режиме, название которого «Единичный режим». Единичный режим позволяет запрашивать и получать информацию только по одному отправлению [17].

При работе с API сервиса отслеживания почтовых отправлений имеются количественные ограничения, которые снимаются при наличии договора с Почтой России и регистрации клиент как «федерального». Без договора количественные ограничения – следующие: максимальное суточное количество единичных запросов – 100 штук.

* 1. **Работа с программным средством**

Интерфейс программы для работы с почтовыми отправлениями показан на рисунке 4.1.

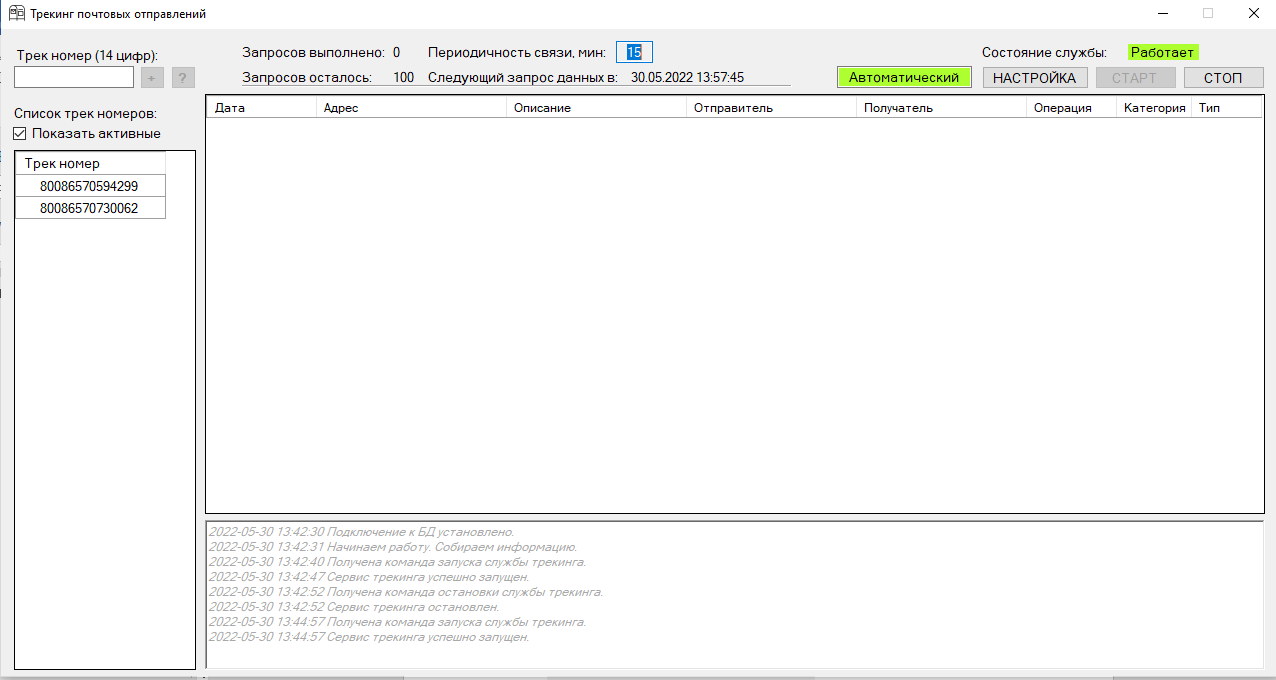


Рисунок 4.1 – Интерфейс окна «Трекинг почтовых отправлений»

Программа для работы с почтовыми отправлениями имеет гибкие возможности настройки Сервиса. В верхней правой части программы имеется возможность управления остановкой и запуском службы Post Tracking Service с помощью кнопок «СТАРТ» и «СТОП». Кроме этого, отображается статус службы: «Остановлена» или «Работает» как показано на рисунках 4.2-4.3.

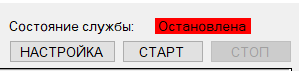


Рисунок 4.2 – Состояние работы службы «Остановлена»



Рисунок 4.3 – Состояние работы службы «Работает»

При нажатии на кнопку «НАСТРОЙКА» происходит открытие второго окна «Настройка времени опроса», которое служит для настройки времени обращений к сервису отслеживания Почты России. Интерфейс окна «Настройка времени опроса» показан на рисунке 4.4.

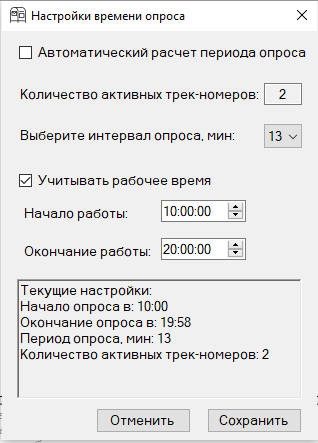


Рисунок 4.4 – Интерфейс окна «Настройка времени опроса»

Для того, чтобы время отправления запросов рассчиталось самостоятельно необходимо установить галочку возле поля «Автоматический расчет периода опроса», в этом случае выбор интервала опроса будет невозможен, рисунок 4.5. Режим расчета времени отображается в окне «Трекинг почтовых отправлений», рисунок 4.6. Для учета рабочего времени необходимо установить галочку возле поля «Учитывать рабочее время» в противном случае установка рабочего времени будет недоступна, рисунок 4.7.

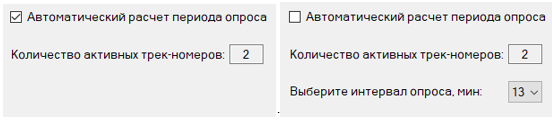


Рисунок 4.5– Выбор автоматического расчета опроса

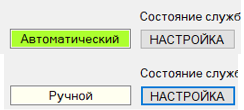


Рисунок 4.6 – Режим расчета времени

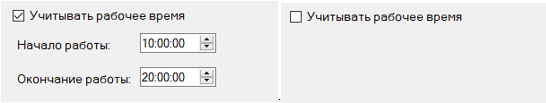


Рисунок 4.7 – Выбор рабочего времени

При установке неверного временного интервала, пользователя проинформируют об ошибке, рисунок 4.8.

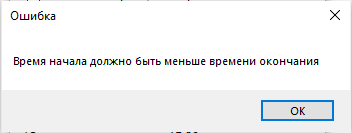


Рисунок 4.8 – Информация об ошибке

Выбор интервала опроса происходит с помощью выпадающего списка как показано на рисунке 4.9.

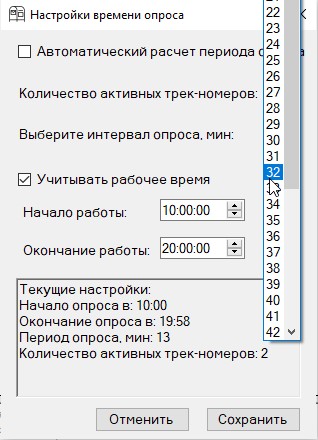


Рисунок 4.9– Выбор интервала опроса

Изменение времени работы происходит с помощью стрелочек или цифр на клавиатуре. Все текущие настройки отображаются в нижней части окна, рисунок 4.10.

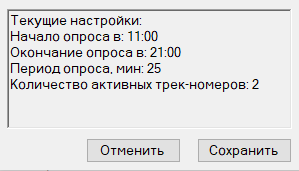


Рисунок 4.10 – Настройки времени опроса

Для сохранения выбранных настроек необходимо нажать кнопку «Сохранить». Для отмены сохранения измененных настроек необходимо нажать кнопку «Отменить» или закрыть окно без сохранения настроек.

В верхней части окна «Трекинг почтовых отправлений» отображается информация по количеству отправленных и оставшихся запросов, времени следующего запроса и периодичности связи, рисунок 4.11.



Рисунок 4.11 – Информация о запросах

В верхней левой части окна «Трекинг почтовых отправлений» производится добавление в БД трек-номеров и их поиск. Трек-номер состоит из 14 цифр и вводится в поле с надписью «Трек-номер (14 цифр)». Добавление производится при нажатии на кнопку с символом «+», поиск производится при нажатии на кнопку с символом «?», рисунок 4.12. При попытке поиска трек-номера несуществующего в БД появится сообщение об ошибке, как показано на рисунке 4.13.



Рисунок 4.12 – Поле ввода трек-номера

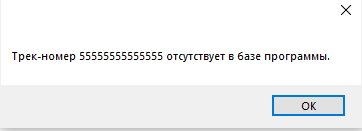


Рисунок 4.13 – Сообщение об ошибке

Список трек-номеров находится в левой части окна «Трекинг почтовых отправлений». Есть возможность показать только трек-номера, по которым отправляются запросы, или показать все добавленные трек-номера. Для этого необходимо установить галочку рядом с полем «Показать активные», рисунок 4.14.

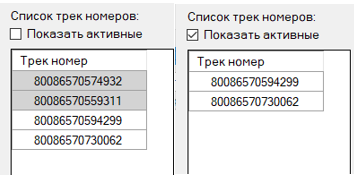


Рисунок 4.14 – Список трек-номеров

При нажатии ПКМ по трек-номеру появляется контекстное меню, которое позволяет отправить запрос по этому трек-номеру сервису отслеживания Почты России, показать операции по отправлению, сохраненные в БД, или сделать трек-номер не активным, рисунок 4.15.

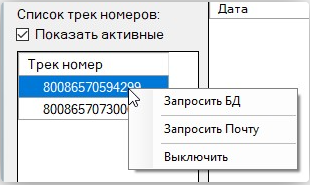


Рисунок 4.15 – Контекстное меню

В центре окна «Трекинг почтовых отправлений» располагается таблица, которая отображает информацию с операциями по отправлению, рисунок 4.16.



Рисунок 4.16 – Таблица с информацией по отправлению

В нижней части окна «Трекинг почтовых отправлений» располагается информация о работе приложения, рисунок 4.17

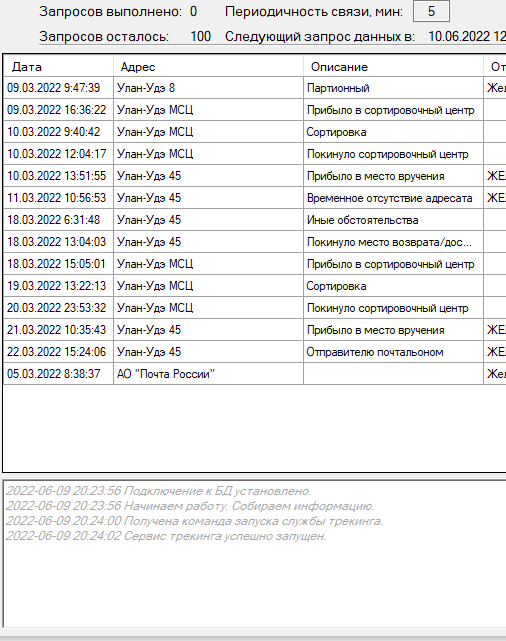


Рисунок 4.17 – Окно с информацией о работе

Для реализованного программного средства разработано руководство пользователя. Описан весь функционал работы ПО, а именно работа с вводом и поиском данных, информирование об ошибках при вводе некорректных данных, отображение информации, настройка времени отправления запросов и запуск/остановка службы. Изучив данное руководство, пользователь получает всю информацию о работе с программным средством.

# **ЗАКЛЮЧЕНИЕ**

Целью данной выпускной квалификационной работы была разработка службы Windows для интеграции с API Сервиса отслеживания почтовых отправлений Почты России, которая позволила автоматизировать процесс отслеживания информации об операциях, совершенных с почтовыми отправлениями.

В рамках выпускной квалификационной работы все поставленные задачи были решены:

* изучено понятие и назначение служб Windows;
* изучено понятие интерфейсов прикладного программирования (API);
* проведено ознакомление c API Сервиса отслеживания почтовых отправлений Почты России;
* произведено моделирование процессов ПО;
* разработана служба Windows для интеграции с API Сервиса отслеживания постовых отправлений Почты России;
* разработано приложение для управления службой и базой данных;
* протестировано разработанное ПО.

Все поставленные задачи выполнены в полном объеме. Цель выпускной квалификационной работы достигнута.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Technologies Consulting. Windows Services Frequently Asked Questions. [Электронный ресурс]. URL: <https://www.coretechnologies.com/WindowsServices/FAQ.html> (дата обращения: 28.03.2022).
2. Lifewire TECH FOR HUMANS. What Is a Service? [Электронный ресурс]. URL: <https://www.lifewire.com/what-is-a-service-4107276> (дата обращения: 28.03.2022).
3. Blog Postman. Intro to APIs: What Are APIs Used for? [Электронный ресурс]. URL: <https://blog.postman.com/intro-to-apis-what-are-apis-used-for/> (дата обращения: 12.04.2022).
4. AltexSoft. What is API: Definition, Types, Specifications, Documentation. [Электронный ресурс]. URL: <https://www.altexsoft.com/blog/engineering/what-is-api-definition-types-specifications-documentation/> (дата обращения: 22.04.2022).
5. The RapidAPI Blog. API Integration – What is API Integration? [Электронный ресурс]. URL: <https://rapidapi.com/blog/api-glossary/api-integration/> (дата обращения: 22.04.2022).
6. Почта России. API. трекинга [Электронный ресурс]. URL: https://tracking.pochta.ru/specification (дата обращения: 22.04.2022).
7. Почта России. API. Интеграция с вашими приложениями [Электронный ресурс]. URL: https://www.pochta.ru/support/business/api (дата обращения: 22.04.2022).
8. UML – Аналитик. [Электронный ресурс]. URL: https:// www.sites.google.com/site/infoprobusinessanalysis/project-definition/uml (дата обращения: 18.05.2022).
9. Шеер Моделирование бизнес-процессов / Шеер, Август-Вильгельм. - М.: Серебряные нити, 2017. - 219 c.
10. ИНТУИТ. Лекция 8: Элементы графической нотации диаграммы последовательности. [Электронный ресурс]. URL: https://intuit.ru/studies/courses/32/32/lecture/1014 (дата обращения: 18.05.2022).
11. ИНТУИТ. Лекция 9: Элементы графической нотации диаграммы состояний. [Электронный ресурс]. URL: https://intuit.ru/studies/courses/32/32/lecture/1016 (дата обращения: 18.05.2022).
12. Брайан, Джонсон Основы Microsoft Visual Studio .NET 2003 / Джонсон Брайан. - М.: Русская Редакция, 2003. - 214 c.
13. Microsoft документация. Новые возможности Visual Studio 2019 [Электронный ресурс]. URL: https://docs.microsoft.com/ru-ru/visualstudio/ide/whats-new-visual-studio-2019?view=vs-2019 (дата обращения: 10.05.2022).
14. Jonathan Crozier. How to debug a .NET Windows Service from Visual Studio [Электронный ресурс]. URL: https://jonathancrozier.com/blog/how-to-debug-a-dot-net-windows-service-from-visual-studio (дата обращения: 11.05.2022).
15. METANIT. Установка службы. [Электронный ресурс]. URL: https://metanit.com/sharp/tutorial/21.2.php (дата обращения: 12.05.2022).
16. Microsoft документация. Функциональная классификация элементов управления Windows Forms. [Электронный ресурс]. URL: https://docs.microsoft.com/ru-ru/dotnet/desktop/winforms/controls/windows-forms-controls-by-function?view=netframeworkdesktop-4.8 (дата обращения: 11.05.2022).
17. Корпоративные хранилища данных. Интеграция систем. Проектная документация. РД 50-34.698-90 Руководство пользователя. [Электронный ресурс]. URL: https://www.prj-exp.ru/patterns/pattern\_user\_guide.php (дата обращения: 13.05.2022).