МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФГАОУ ВО »СЕВАСТОПОЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Кафедра «Прикладная математика
и информатика»

ОТЧЕТ

о выполнении индивидуального задания № 3
по дисциплине
»ЦИФРОВЫЕ ТЕХНОЛОГИИ И МЕДИАБЕЗОПАСНОСТЬ»
Специализированная мастерская «ЦИФРОВОЙ ОФИС.
ПРОМТ-ИНЖИНИРИНГ ДЛЯ ПОВСЕДНЕВНЫХ ЗАДАЧ»
Вариант № 1

Выполнил:
студент группы ИИ/б-25-6-о
Заварзин А.В.

Проверили:
доцент кафедры ПМиИ
Ченгарь О.В.

Севастополь, 2025

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

ЦЕЛЬ РАБОТЫ: научиться применять техники промт-инжиниринга для генерации кода на языке PlantUML, позволяющего создавать UMLдиаграммы без наличия предварительных знаний в области моделирования систем.

ЗАДАЧА: создать промпты для генеративных ИИ, которые позволят на основе текстового описания одной системы последовательно создать промты для генерации кода следующих диаграмм: прецедентов (Use Case), последовательности (Sequence), классов (Class), деятельности (Activity) и состояний (State).

ХОД ВЫПОЛНЕНИЯ РАБОТЫ

ШАГ 1: Подготовка и описание системы для диаграммы UseCase

Система для диаграммы UseCase по угрозе «Фишинг» будет иметь следующий вид:

1. Акторы (Действующие лица):
   - 3.1 «Сотрудник»
   - 3.2 «Администратор»
   - 3.3 «Система анализа»
2. Основные сущности (Объекты):
   - 2.1 Электронное письмо имеет атрибуты (отправитель, тема, статус)
   - 2.2 Отчет об инциденте имеет атрибуты (ID, уровень угрозы, описание)
   - 2.3 Правило фильтрации имеет атрибуты (условие, действие)
3. Базовые процессы и сценарии:
   - 3.1 Сотрудник может «Получать письма» и «Сообщать о фишинге»
   - 3.2 Администратор «Управляет правилами», «Просматривает отчеты»
   - 3.3 Система «Анализирует письма» (процесс включает проверку ссылок и текста, вынесение вердикта) и «Формирует отчеты»
4. Жизненный цикл: «Отчет об инциденте» проходит состояния: «Новый» → «В обработке» → «Закрыт».

Эта простая система создает замкнутый цикл безопасности: сотрудник обнаруживает угрозу, система ее автоматически анализирует и классифицирует, а администратор принимает финальное решение, постоянно улучшая правила на основе поступающих отчетов.

ШАГ 2: Генерация диаграммы прецедентов (Use Case)

Опираясь на систему для диаграммы UseCase в предыдущем шаге, был составлен промт для генерирования диаграммы прецедентов для нейросети DeepSeek:

```
Напиши качественный промт для нейросети, основываясь на этой информации: Ты квалифицированный специалист и профессионал в сфере UML-диаграмм, PlantUML. По теме «Системы обнаружения фишинга» напиши кратко информацию для диаграммы прецедентов Use-Case и напиши код на PlantUML для неё.

Контекст системы:

Акторы: «Сотрудник», «Администратор», «Система анализа».

Сущности: «Электронное письмо» (атрибуты: отправитель, тема, статус),»Отчет об инциденте» (ID, уровень угрозы), «Правило фильтрации».

Процессы и сценарии: Сотрудник «Получает письма» и «Сообщает о фишинге».

Администратор «Управляет правилами», «Просматривает отчеты». Система

«Анализирует письма», «Формирует отчеты».

Жизненный цикл: «Отчет об инциденте» проходит состояния: «Новый» -> «В

обработке» -> «Закрыт».
```

Полученный промт был загружен в нейросеть DeepSeek. DeepSeek сгенерировал следующий код для PlantUMLи описание диаграммы:

```
@startuml
left to right direction


actor "Сотрудник" as Employee
actor "Администратор" as Admin
actor "Система анализа" as System


rectangle "Модуль обнаружения фишинга" {
 usecase "Получает письма" as ReceiveEmails
 usecase "Сообщить о фишинге" as ReportPhishing


 usecase "Управлять правилами фильтрации" as ManageRules
```

```
 usecase "Просматривать отчеты об инцидентах" as ViewReports


 usecase "Анализировать письма" as AnalyzeEmails
 usecase "Формировать отчет об инциденте" as GenerateReport
}


Employee --> ReceiveEmails
Employee --> ReportPhishing


Admin --> ManageRules
Admin --> ViewReports


System --> AnalyzeEmails
System --> GenerateReport


' Автоматический анализ при получении письма
ReceiveEmails ..> AnalyzeEmails : include


' Формирование отчета при ручном отчете и автоматическом анализе
ReportPhishing ..> GenerateReport : include
AnalyzeEmails ..> GenerateReport : include


@enduml
```

Текстовое описание диаграммы, полученное вместе с кодом PlantUML от нейросети DeepSeek:

Данная диаграмма прецедентов описывает модуль "Системы обнаружения фишинга" для корпоративной почты. Система обеспечивает автоматический анализ входящих писем, обработку ручных жалоб от сотрудников и управление правилами фильтрации администратором. Ключевые акторы включают Сотрудника (получение писем и отчет о фишинге), Администратора (управление правилами и просмотр отчетов) и

Систему анализа (автоматический анализ и формирование отчетов). Прецеденты связаны через отношения включения: анализ писем выполняется автоматически при получении, а отчет об инциденте формируется как при автоматическом обнаружении, так и при ручном сообщении о фишинге.

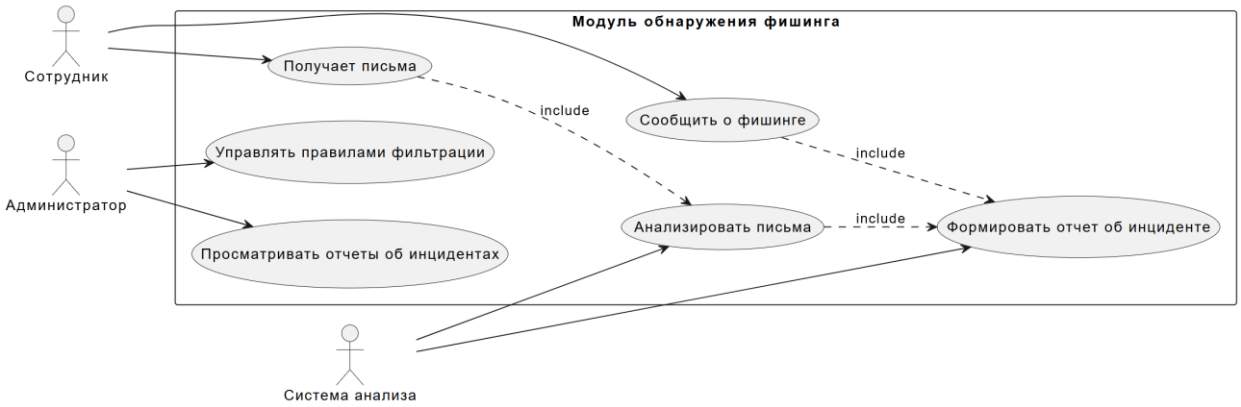

Рисунок 2.1 – Диаграмма UseCase для угрозы «Фишинг-письма»

В результате реализации вышеприведённого кода PlantUML была получена следующая диграмма, представленная на Рисунке 2.1

ШАГ 3: Диаграмма классов (Class)

Опираясь на систему для диаграммы UseCase в предыдущем шаге, был составлен промт для генерирования диаграммы прецедентов для нейросети DeepSeek:

```
Напиши качественный промт для нейросети, основываясь на этой информации: Ты квалифицированный специалист по кибербезопасности и профессионал в сфере UML-диаграмм, PlantUML. По теме «Системы обнаружения фишинга» напиши кратко информацию для диаграммы классов (Class Diagram) и напиши код на PlantUML для неё.

Контекст системы:

Акторы: «Сотрудник», «Администратор», «Система анализа».

Сущности: «Электронное письмо» (атрибуты: отправитель, тема, статус),»Отчет об инциденте» (ID, уровень угрозы), «Правило фильтрации».

Процессы и сценарии: Сотрудник «Получает письма» и «Сообщает о фишинге».

Администратор «Управляет правилами», «Просматривает отчеты». Система
```

```
«Анализирует письма», «Формирует отчеты».
Жизненный цикл: «Отчет об инциденте» проходит состояния: «Новый» -> «В
обработке» -> «Закрыт».
```

Полученный промт был загружен в нейросеть DeepSeek. DeepSeek сгенерировал следующий код для PlantUMLи описание диаграммы:

```
@startuml

' Определение основных классов системы
class Пользователь {
 # id: Long
 # имя: String
 # email: String
 # датаРегистрации: Date
 + аутентифицировать()
 + получитьПрофиль()
}

class Сотрудник {
 - отдел: String
 - должность: String
 + получатьПисьма()
 + сообщитьОФишинге()
}

class АдминистраторБезопасности {
 - уровеньДоступа: Integer
 + управлятьПравиламиФильтрации()
 + просматриватьОтчеты()
}

class Письмо {
```

```
 - id: Long
 - отправитель: String
 - тема: String
 - содержание: String
 - датаПолучения: Date
 - хэш: String
 - меткаФишинг: Boolean
 + проверятьФишинг()
 + помечатьКакФишинг()
}

class ОтчетОбИнциденте {
 - id: Long
 - датаСоздания: Date
 - статус: String {Новый, В обработке, Закрыт}
 - типОбнаружения: String {Авто, Ручной}
 - описание: String
 + создаватьОтчет()
 + обновлятьСтатус()
 + вычислятьСтатистику()
}

class ПравилоФильтрации {
 - id: Long
 - название: String
 - условие: String
 - действие: String
 - приоритет: Integer
 - активен: Boolean
 + активировать()
 + деактивировать()
```

```
 + проверятьПравило()
}

class СистемаАнализа {
 - id: Long
 - версия: String
 - настройки: String
 + анализироватьПисьма()
 + формироватьОтчет()
 + обучатьМодель()
}

' Определение отношений наследования
Пользователь <|-- Сотрудник
Пользователь <|-- АдминистраторБезопасности

' Определение основных ассоциаций
Сотрудник "1" -- "*" Письмо : получает
Сотрудник "1" -- "*" ОтчетОбИнциденте : создает

АдминистраторБезопасности "1" -- "*" ПравилоФильтрации : управляет
АдминистраторБезопасности "1" -- "*" ОтчетОбИнциденте : просматривает

Письмо "1" -- "0..1" ОтчетОбИнциденте : анализируется в
СистемаАнализа "1" -- "*" Письмо : анализирует
СистемаАнализа "1" -- "*" ОтчетОбИнциденте : формирует
ПравилоФильтрации "1" -- "*" Письмо : фильтрует

' Композиция - Правила являются частью Системы анализа
СистемаАнализа *-- "1..*" ПравилоФильтрации : содержит
```

```
' Агрегация - Отчеты могут существовать независимо от администраторов

АдминистраторБезопасности o-- "0..*" ОтчетОбИнциденте


@enduml
```

Текстовое описание диаграммы, полученное вместе с кодом PlantUML от нейросети DeepSeek:

Диаграмма классов модуля "Системы обнаружения фишинга" представляет ключевые сущности и их взаимосвязи. Основные классы включают Пользователь (абстрактный класс с наследниками Сотрудник и Администратор), Письмо (содержит информацию о входящем сообщении), ОтчетОбИнциденте (отслеживает статус обработки фишинга), ПравилоФильтрации (определяет критерии анализа) и СистемаАнализа (обрабатывает письма). Отношения между классами отражают бизнес-логику: система анализирует письма, создает отчеты, а администраторы управляют правилами фильтрации.

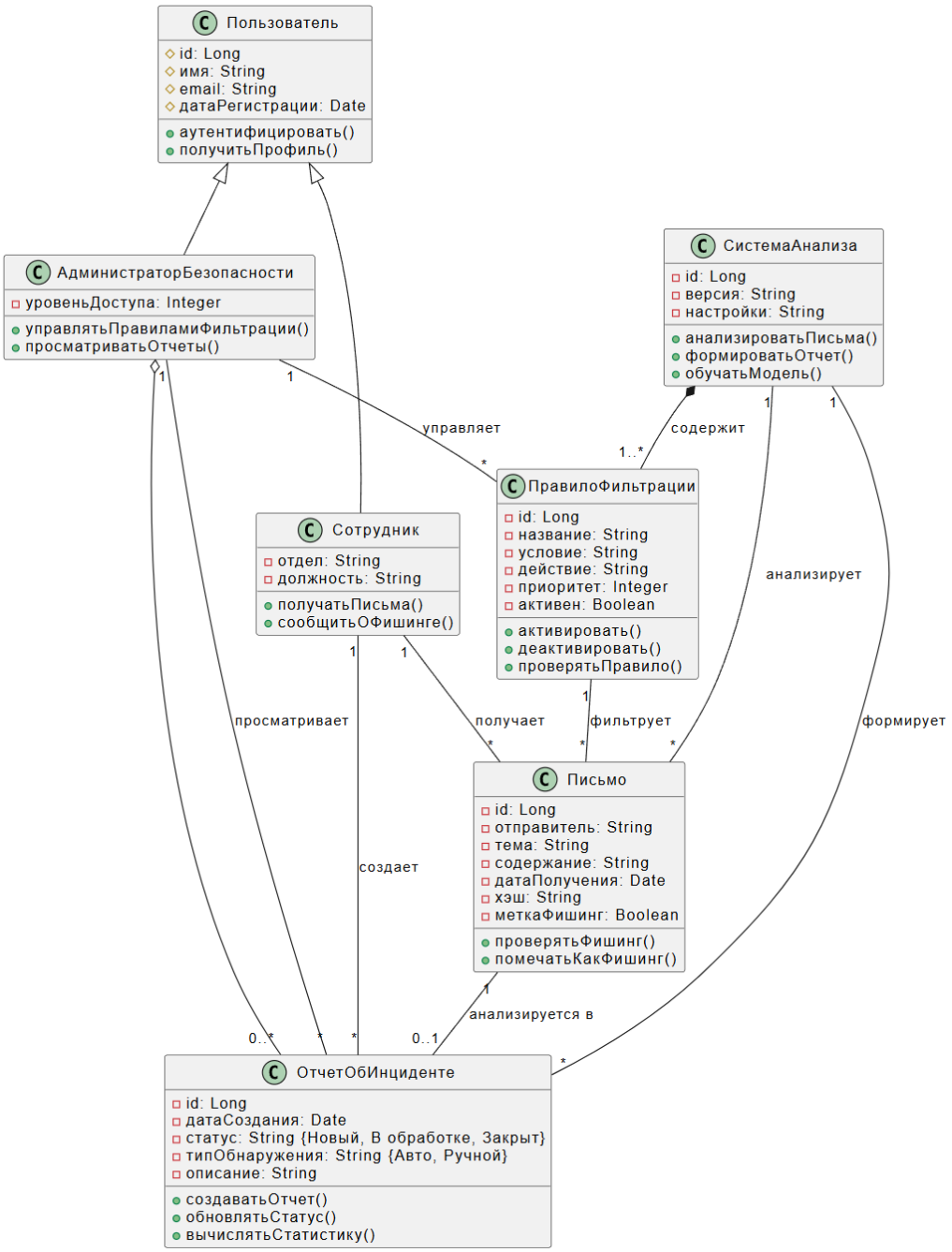

Рисунок 3.1 – Диаграмма Class для угрозы «Фишинг-письма»

В результате реализации вышеприведённого кода PlantUML была получена следующая диграмма, представленная на Рисунке 3.1

ШАГ 4: Диаграмма деятельности (Activity)

Для получения промта для генерации диаграммы деятельности был сформирован следующий промт для нейросети DeepSeek:

```
Напиши мне качественный промт для нейросети, основываясь на этой информации: "Ты квалифицированный системный аналитик. По теме "Системы обнаружения фишинга" напиши краткую информацию для диаграммы деятельности и напиши код на PlantUML для неё.

Контекст системы:

Акторы: "Сотрудник", "Администратор", "Система анализа".

Сущности: "Электронное письмо" (атрибуты: отправитель, тема, статус), "Отчет об инциденте" (ID, уровень угрозы), "Правило фильтрации".

Процессы и сценарии: Сотрудник "Получает письма" и "Сообщает о фишинге". Администратор "Управляет правилами", "Просматривает отчеты". Система "Анализирует письма", "Формирует отчеты".

Жизненный цикл: "Отчет об инциденте" проходит состояния: "Новый" -> "В обработке" -> "Закрыт".

Подробное описание процесса "Анализ входящего письма":

1) Процесс начинается.

2) Происходит действие "Получить письмо".

3) Далее "Извлечь ссылки и вложения".

4) Затем "Проверить по базе угроз".

5) После проверки идет условие: "Найдена угроза?".

6) Если да, то выполняется действие "Отправить в карантин"; Если нет, то выполняется "Доставить сотруднику".

7) Процесс завершается."
```

Полученный промт был загружен в нейросеть DeepSeek. DeepSeek сгенерировал следующий код для PlantUMLи описание диаграммы:

```
@startuml

|Получение данных|

start

:Получить письмо;

:Извлечь ссылки и вложения;


|Анализ угроз|

:Проверить по базе угроз;
```

```
|Обработка результата|
if (Найдена угроза?) then (да)
 :Отправить в карантин;
else (нет)
 :Доставить сотруднику;
endif
stop
@enduml
```

Текстовое описание диаграммы, полученное вместе с кодом PlantUML от нейросети DeepSeek:

Процесс начинается с получения письма системой. Далее извлекаются все ссылки и вложения для анализа. На основном этапе проверки письмо сравнивается с базой известных угроз. В точке принятия решения система определяет, была ли обнаружена угроза. Если угроза найдена, письмо перемещается в карантин. Если угроз нет, письмо доставляется сотруднику. Процесс завершается после выполнения одного из этих действий.

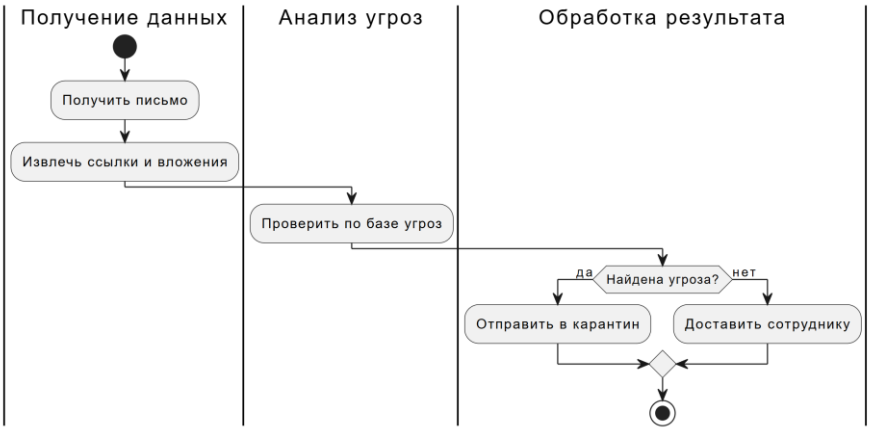

Рисунок 4.1 – Диаграмма Activity для угрозы «Фишинг-письма»

В результате реализации вышеприведённого кода PlantUML была получена следующая диграмма, представленная на Рисунке 4.1

ШАГ 5: Диаграмма последовательности (Sequence)

Для получения промта для генерации диаграммы последовательности был сформирован следующий промт для нейросети DeepSeek:

```
Напиши мне качественный промт для нейросети, основываясь на этой информации: "Ты квалифицированный специалист по компьютерной безопасности. По теме "Системы обнаружения фишинга" напиши краткую информацию для диаграммы последовательности и напиши код на PlantUML для неё.

Контекст системы:

Акторы: "Сотрудник", "Администратор", "Система анализа".

Сущности: "Электронное письмо" (атрибуты: отправитель, тема, статус), "Отчет об инциденте" (ID, уровень угрозы), "Правило фильтрации".

Процессы и сценарии: Сотрудник "Получает письма" и "Сообщает о фишинге". Администратор "Управляет правилами", "Просматривает отчеты". Система "Анализирует письма", "Формирует отчеты".

Жизненный цикл: "Отчет об инциденте" проходит состояния: "Новый" -> "В обработке" -> "Закрыт".
```

Полученный промт был загружен в нейросеть DeepSeek. DeepSeek сгенерировал следующий код для PlantUMLи описание диаграммы:

```
@startuml
' Диаграмма последовательности обработки фишингового инцидента
title Сценарий "Обработка подозрительного письма и эскалация инцидента"

actor "Сотрудник" as Employee
actor "Администратор" as Admin
participant "Система анализа" as System

' Начало сценария: сотрудник получает письмо
Employee -> System: Сообщить о подозрительном письме\n(отправитель, тема)
note right of Employee: Письмо получено, статус: не проверен

' Система анализирует письмо
System -> System: Анализ письма по правилам фильтрации
```

```
      note right of System: Проверка отправителя и темы\nна соответствие фишинговым шаблонам


      ' Создание отчета об инциденте
      System -> System: Создать отчет об инциденте\n(ID, уровень угрозы)
      note right of System: Статус отчета: "Новый"


      ' Уведомление администратора
      System -> Admin: Уведомить о новом инциденте


      ' Администратор обрабатывает инцидент
      Admin -> System: Просмотреть отчет об инциденте
      System --> Admin: Возврат деталей отчета и письма
      Admin -> System: Изменить статус отчета на "В обработке"
      note right of Admin: Статус отчета обновлен


      ' Дополнительные действия администратора
      Admin -> System: Обновить правила фильтрации (при необходимости)
      note right of Admin: Блокировка подобных угроз в будущем


      ' Завершение инцидента
      Admin -> System: Закрыть инцидент\n(изменить статус на "Закрыт")
      System --> Admin: Подтверждение закрытия
      note right of System: Статус письма и отчета обновлен


      @enduml
```

Текстовое описание диаграммы, полученное вместе с кодом PlantUML от нейросети DeepSeek:

Сотрудник получает подозрительное электронное письмо и инициирует процесс, сообщая о фишинге через систему анализа. Система анализа принимает отчет, изменяет статус письма на "Анализируется" и начинает

автоматическую проверку содержимого по существующим правилам фильтрации. После анализа система создает новый отчет об инциденте с уровнем угрозы и устанавливает его статус "Новый". Администратор получает уведомление о новом инциденте и переводит отчет в статус "В обработке" для дальнейшего расследования. Администратор проверяет детали письма и отчета, при необходимости обновляет правила фильтрации для блокировки подобных угроз. По завершению обработки администратор закрывает инцидент, меняя статус отчета на "Закрыт". Система фиксирует изменения и обновляет статус связанного письма. Весь процесс отслеживается в системе, обеспечивая прозрачность и документирование действий.

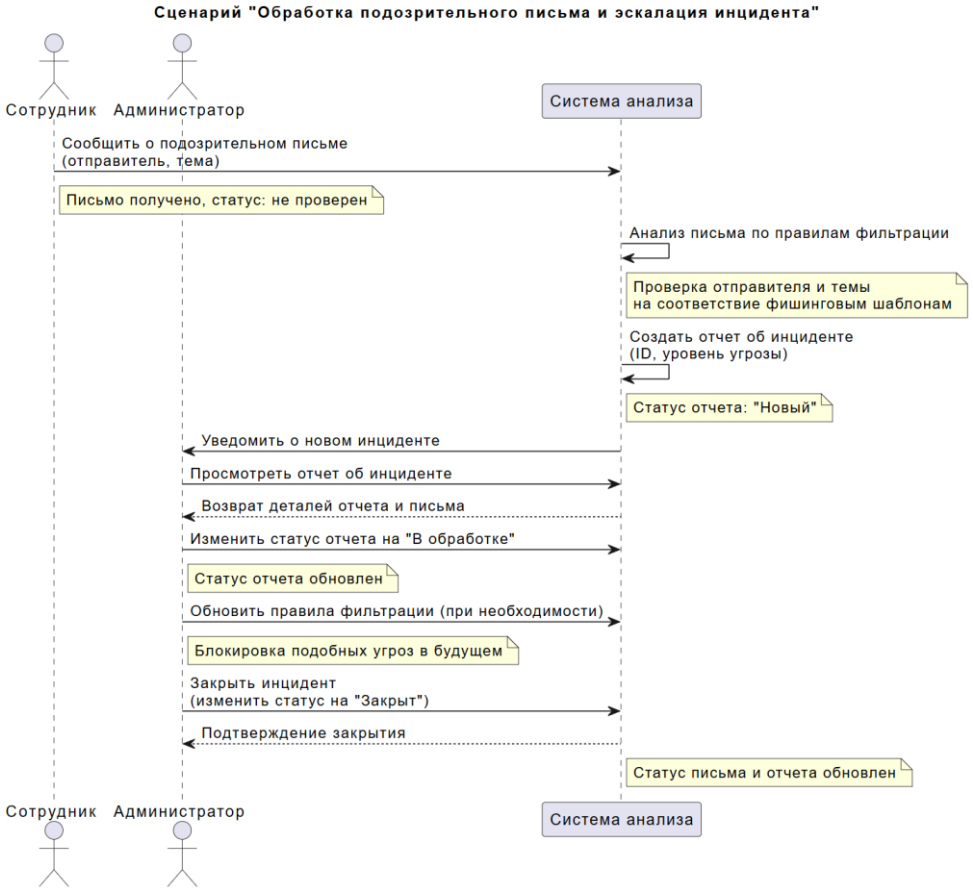

Рисунок 4.1 – Диаграмма Sequence для угрозы «Фишинг-письма»

В результате реализации вышеприведённого кода PlantUML была получена следующая диграмма, представленная на Рисунке 4.1

ВЫВОДЫ ПО РАБОТЕ

В ходе работы был разработан комплекс промптов для генеративных ИИ, позволяющих на основе текстового описания системы получать код на языке PlantUML для четырёх типов UML-диаграмм. Последовательное создание промтов для диаграмм прецедентов, последовательности, классов и деятельности подтвердило возможность эффективного применения техник промт-инжиниринга для автоматизации процесса моделирования. Это позволило достичь цели работы — продемонстрировать возможность создания UML-диаграмм без необходимости глубоких предварительных знаний в области моделирования систем.

Таким образом, применение структурированных промптов показало свою эффективность для автоматизированной генерации графических моделей. Результаты работы свидетельствуют о том, что данный подход существенно снижает порог входа для создания UML-диаграмм, предоставляя специалистам из различных предметных областей инструмент для визуализации архитектуры систем. В ходе работы были получены практические навыки составления и оптимизации промптов для решения конкретной задачи генерации кода для построения диаграмм PlantUML.