**ТЕХНИЧЕСКИЙ ПРОЕКТ**  
«Telegram-бот для просмотра информации о рыночных активах»

Киров,

2024

**Содержание**

[**1.** **Концепция предлагаемого решения** 3](#_Toc182711433)

[**2.** **Пользовательский интерфейс** 3](#_Toc182711434)

[**3.** **Уточнение структуры данных** 10](#_Toc182711435)

[**3.1.** **Описание основных сущностей** 10](#_Toc182711436)

[**3.2.** **Построение логической ER-диаграммы** 12](#_Toc182711437)

[**3.3.** **Построение физической ER-диаграммы** 12](#_Toc182711438)

[**3.4.** **Построение диаграммы потоков данных** 13](#_Toc182711439)

[**4.** **Бизнес-процессы системы** 14](#_Toc182711440)

[**5.** **Разработка алгоритма решения задачи** 16](#_Toc182711441)

[**5.1.** **Алгоритм взаимодействия с пользователем** 16](#_Toc182711442)

[**5.2.** **Алгоритм регистрации пользователя** 17](#_Toc182711443)

[**5.3.** **Алгоритм получения данных о задаче** 18](#_Toc182711444)

[**5.4.** **Алгоритм настройки уведомлений** 19](#_Toc182711445)

[**5.5.** **Алгоритм отправки уведомлений** 20](#_Toc182711446)

[**6.** **Выбранная архитектура системы** 21](#_Toc182711447)

[**6.1. Общая архитектура** 21](#_Toc182711448)

[**6.2. Технологический стек** 21](#_Toc182711449)

[**6.3. Структура программы** 22](#_Toc182711450)

# **Концепция предлагаемого решения**

Проект представляет собой Telegram-бот, который предоставляет пользователю актуальную информацию о рыночных активах (акциях, криптовалютах, валютных парах). Основные возможности бота включают:

* Получение текущих котировок активов.
* Просмотр исторических данных и графиков изменений цен.
* Управление списком избранных активов.
* Настройка уведомлений по различным условиям, таким как изменение цен или достижение заданного порога.
* Простой и удобный интерфейс на платформе Telegram с использованием текстовых команд и встроенных клавиатур.

# **Пользовательский интерфейс**

Пользовательский интерфейс реализуется через платформу Telegram с использованием текстовых сообщений и встроенных клавиатур (Inline Keyboards).

Макеты интерфейсов:

Главное меню является стартовой точкой взаимодействия с ботом. Оно содержит четыре основные кнопки, каждая из которых ведет к определенному функционалу:

* Информация об активах: предоставляет доступ к данным об акциях, валютных парах и криптовалютах.
* Избранные активы: позволяет управлять списком избранных активов.
* Настройки бота: открывает настройки бота, включая параметры уведомлений, часовой пояс и базовую валюту.
* Справка по командам и помощь: предоставляет информацию о возможностях бота и доступных командах.

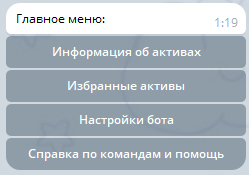


Рисунок 1 – Главное меню

Экран выбора типа актива. Содержит три основные категории:

* Акции: Данные о компаниях и их рыночных котировках.
* Валюты: Информация о курсах валютных пар.
* Криптовалюты: Данные о стоимости популярных криптовалют.
* На экране также доступна кнопка "Назад" для возврата в главное меню.

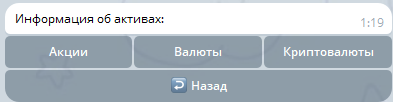


Рисунок 2 – Информация об активе

Экран выбора типа избранных активов. Экран позволяет:

* Просматривать информацию об избранных активах.
* Управлять списком избранного (например, удалять активы). Кнопка "Назад" возвращает пользователя в главное меню.

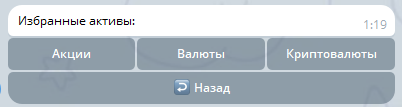


Рисунок 3 – Избранные активы

Экран для изменения параметров взаимодействия с ботом. Включает:

* + Настройку часового пояса.
  + Выбор базовой валюты.
  + Частоту уведомлений.

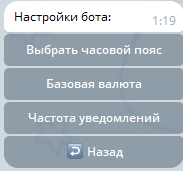


Рисунок 4 – Настройки бота

Экран помощи предоставляет пользователю справочную информацию о возможностях бота.  
Включает описание доступных команд и функций, таких как:

* Просмотр информации об активах (акции, валюты, криптовалюты).
* Управление избранным.
* Настройки (часовой пояс, базовая валюта, уведомления).
* Возможность получения графиков и аналитики.
* Кнопка "Назад" позволяет вернуться в главное меню.

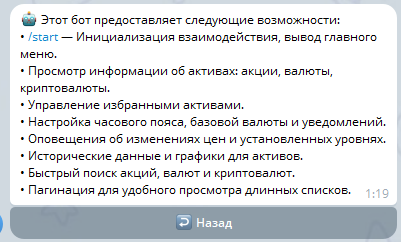


Рисунок 5 – Помощь

Экран акций отображает список доступных акций для просмотра. Пользователь может:

* Выбрать интересующую акцию из списка для получения подробной информации.
* Использовать кнопку "Поиск" для нахождения нужной акции.
* Переключаться между страницами акций, если список длинный.
* Кнопка "Назад" возвращает пользователя к разделу "Информация об активах".

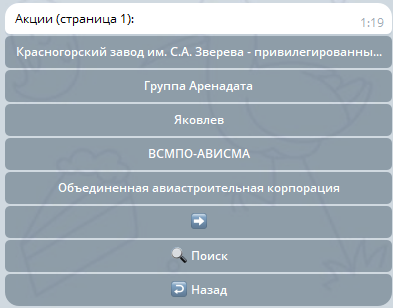


Рисунок 6 – Акции

Экран валют позволяет выбрать валютные пары для анализа. Основные функции:

* Выбор двух валют для сравнения курса.
* Подсветка базовой валюты пользователя (например, RUB).
* Подтверждение выбора через кнопку "Получить информацию".
* Кнопка "Назад" возвращает в меню "Информация об активах".

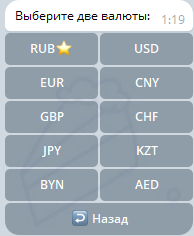


Рисунок 7 – Валюты

Экран криптовалют представляет список популярных криптовалют, доступных для анализа.

* Каждая кнопка криптовалюты ведет к детальной информации о выбранном активе.
* Кнопка "Назад" возвращает в раздел "Информация об активах".
* Список криптовалют разделен по строкам для удобной навигации.



Рисунок 8 – Криптовалюты

Экран сводки об активе предоставляет детализированную информацию о выбранном активе.

* Данные: текущая цена, минимальная и максимальная цена за день, рыночная капитализация.
* Управление активом:
  + График: кнопка для отображения графика изменения цены.
  + Уведомления: настройка уведомлений по изменению цены.
  + Избранное: возможность добавить/убрать актив в/из избранного.
  + Обновление информации: кнопка для получения актуальных данных об активе.

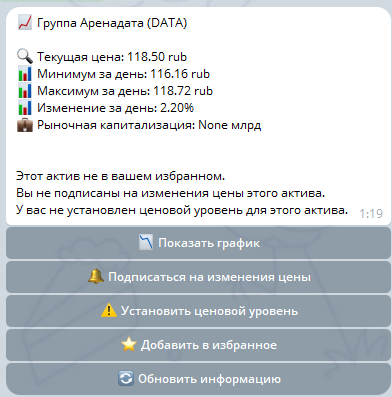


Рисунок 9 – Сводка об активе

Экран избранных активов отображает список активов, которые пользователь добавил в избранное.

* Навигация:
  + Возможность открыть подробную информацию о каждом активе.
  + Кнопка "Назад" возвращает к предыдущему экрану.

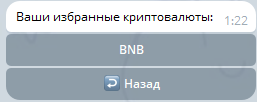


Рисунок 10 – Пример избранных активов

Экран выбора часового пояса отображает доступные часовые пояса, от -12 до +14 часов.

* Отметка текущего часового пояса: активный часовой пояс помечен галочкой.
* Навигация:
  + Кнопка "Назад" возвращает в настройки бота.



Рисунок 11 – Выбор часового пояса

Экран выбора базовой валюты, используемой для отображения данных.

* Особенности:
  + Список включает наиболее популярные валюты (RUB, USD, EUR, и т.д.).
  + Текущая базовая валюта выделена галочкой.
* Навигация:
  + Кнопка "Назад" возвращает в настройки.



Рисунок 12 – Выбор базовой валюты

Экран выбора частоты уведомлений позволяет настроить периодичность уведомлений.

* Функциональность:
  + Возможен выбор интервала между уведомлениями (1 минута, 5 минут, 10 минут и т.д.).
  + Текущая частота уведомлений помечена галочкой.
* Навигация:
  + Кнопка "Назад" возвращает в настройки.



Рисунок 13 – Выбор частоты уведомлений

Экран выбора периода для графика предназначен для настройки временного интервала, на основе которого строится график актива.

* Функциональность:
  + Доступны следующие таймфреймы: "За 1 день", "За 1 неделю", "За 1 месяц", "За 1 год".
* Навигация:
  + Кнопка "Назад" возвращает к предыдущему экрану.

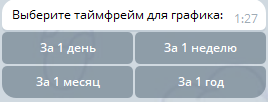


Рисунок 13 – Выбор периода для графика

# **Уточнение структуры данных**

В ходе уточнения данных были выполнены следующие задачи, показанные в Таблице 2.

Таблица 1 – Задачи для уточнения структуры данных.

|  |  |  |  |
| --- | --- | --- | --- |
| **№ задачи** | **Название** | **Содержание** | **Результат** |
| 1 | Определение атрибутов сущностей | Определение и описание атрибутов для каждой сущности, таких как типы данных и ограничения | Таблица с атрибутами сущностей и их описанием |
| 2 | Представление атрибутов сущностей | Создание таблицы с атрибутами для каждой сущности | Готовая таблица с атрибутами сущностей |
| 3 | Построение диаграммы потоков данных | Определить, как будут передвигаться данные в программе; построить DFD диаграмму | Готовая диаграмма потоков данных (DFD) |

* 1. **Описание основных сущностей**

1. User (Пользователь)

Эта сущность описывает пользователей системы, взаимодействующих с Telegram-ботом.

Таблица 2 – Атрибуты таблицы User:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Название** | **Тип** | **Уникальность** | **Обязательность** | **Описание** |
| Id | Integer | Да | Да | Уникальный идентификатор |
| Username | String | Нет | Нет | Имя пользователя в Telegram |

2. Settings (Настройки)

Сущность описывает персональные настройки пользователя, такие как часовой пояс, частота уведомлений и избранные активы.

Таблица 3 – Атрибуты таблицы Settings:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Название** | **Тип** | **Уникальность** | **Обязательность** | **Описание** |
| Id | Integer | Да | Да | Уникальный идентификатор настройки |
| user\_id | Integer | Нет | Да | Ссылка на пользователя |
| settings | JSON | Нет | Да | Настройки пользователя |

Пример структуры JSON для поля settings:

{

 "timezone": "UTC+3",

 "base\_currency": "USD",

 "notification\_frequency": "10min",

 "favorites\_assets": [

  {"asset\_type": "stock", "ticker": "AAPL"},

  {"asset\_type": "crypto", "ticker": "BTC"}

],

 "notification\_price\_change": {

   "stock": ["AAPL", "GOOGL"],

   "crypto": ["BTC", "ETH"]

},

 "notification\_price\_level": {

   "stock": {

     "AAPL": {"type": "value", "value": 150.00, "current\_price": 148.50}

  },

   "crypto": {

     "BTC": {"type": "percent", "value": 5, "current\_price": 45000}

  }

}

}

​Пояснение структуры JSON:

* + timezone: Часовой пояс пользователя.
  + base\_currency: Валюта по умолчанию для отображения цен.
  + notification\_frequency: Частота отправки уведомлений.
  + favorites\_assets: Список избранных активов пользователя.
  + notification\_price\_change: Активы, для которых настроены уведомления об изменении цен.
  + notification\_price\_level: Настройки пороговых уведомлений для активов.

​

* 1. **Построение логической ER-диаграммы**

Сущности:

* Пользователи: включают идентификатор и имя пользователя.
* Настройки: включают идентификатор, ID пользователя и настройки в формате JSON.

Связь: Пользователь может иметь 0 или 1 набор настроек.

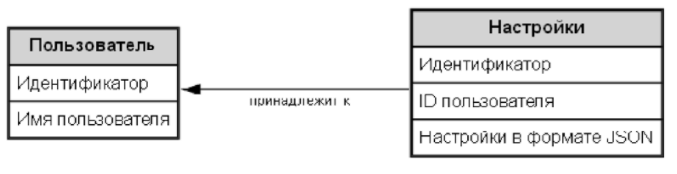
****

Рисунок 14 – Логическая диаграмма

* 1. **Построение физической ER-диаграммы**

Таблица users:

* Поля:
  + id: уникальный идентификатор (INTEGER).
  + username: имя пользователя (VARCHAR).

Таблица settings:

* Поля:
  + id: уникальный идентификатор настройки (INTEGER).
  + settings: JSON с настройками.
  + user\_id: ссылка на пользователя (INTEGER).

Связь между таблицами: users.id ↔ settings.user\_id (0..1 : 0..N).

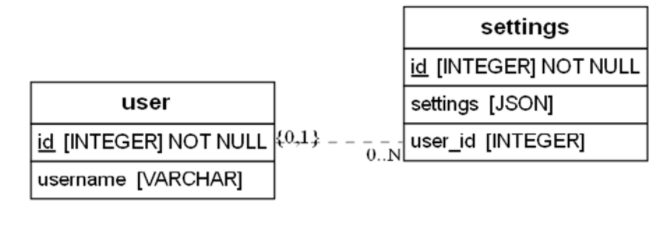
****

Рисунок 15 – Физическая диаграмма

* 1. **Построение диаграммы потоков данных**

Центральный элемент: Telegram-бот.

Потоки:

* Взаимодействие с пользователем (запросы/ответы).
* Работа с данными (запрос/обновление).
* Настройки уведомлений.
* Обработка данных.

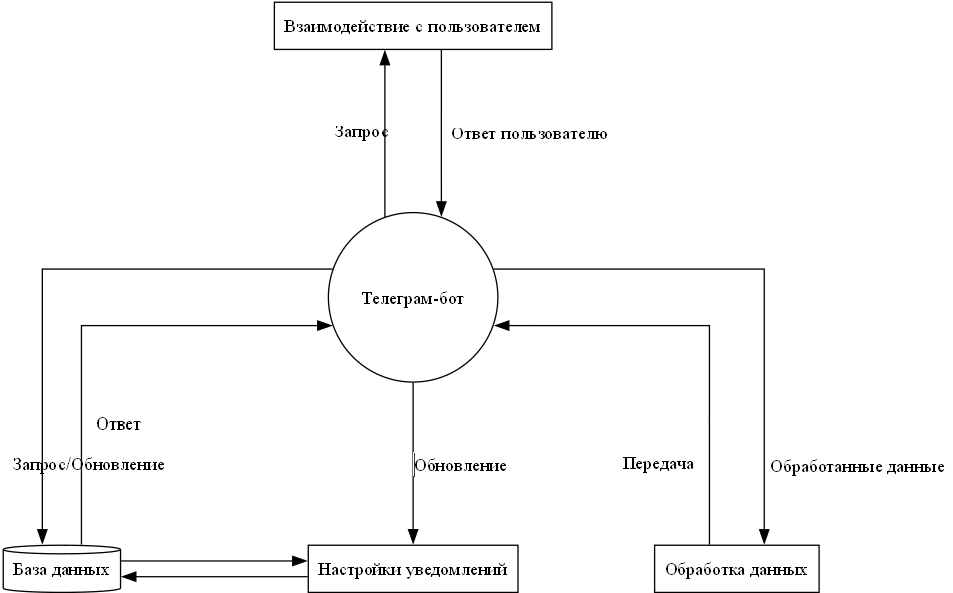
****

Рисунок 16 – Диаграмма потоков данных

# **Бизнес-процессы системы**

На рисунке 14 представлена контекстная диаграмма IDEF0.

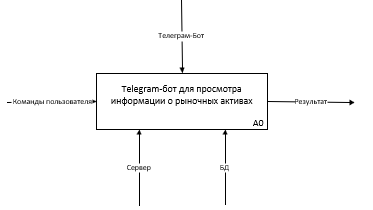


Рисунок 17 – диаграмма IDEF0

Основные бизнес-процессы:

1. Регистрация пользователя и инициализация настроек: при первом взаимодействии с ботом пользователь добавляется в базу данных, и для него создаются базовые настройки.
2. Просмотр информации об активах:
   * + Пользователь выбирает тип актива (акции, валюты, криптовалюты).
     + Бот предоставляет список доступных активов или поиск по ним.
     + При выборе актива отображается текущая цена, исторические данные и другая информация.
3. Управление избранными активами:
   * + Добавление активов в избранное.
     + Удаление активов из избранного.
     + Просмотр списка избранных активов.
4. Настройка уведомлений и оповещений:
   * + Подписка на изменения цен конкретных активов.
     + Установка ценовых уровней для оповещений.
     + Настройка частоты уведомлений.
5. Настройка параметров бота:
   * + Изменение часового пояса.
     + Выбор базовой валюты.
     + Настройка частоты уведомлений.
6. Обработка пользовательского ввода и ошибок:
   * + Валидация вводимых данных.
     + Предоставление пользователю соответствующих сообщений об ошибках или подтверждениях.

На рисунке 15 показан результат декомпозиции деятельности бота.

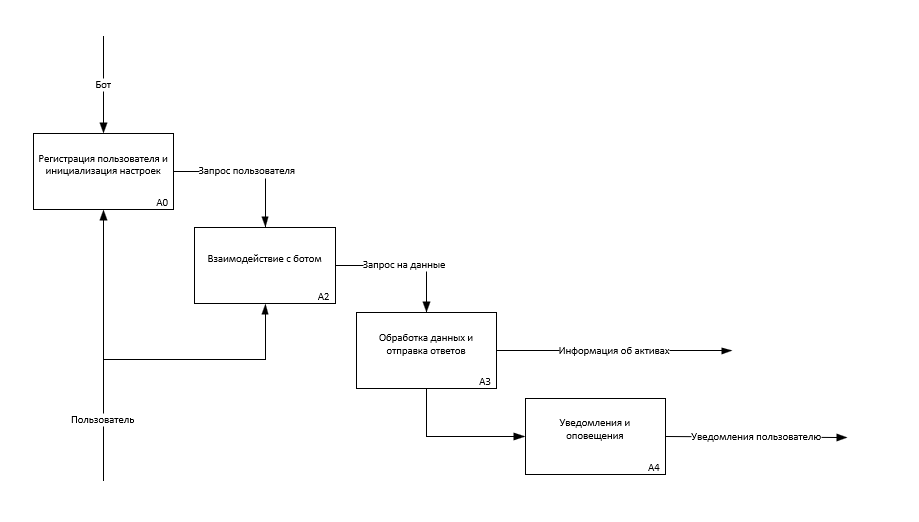


Рисунок 18 – результат декомпозиции деятельности бота для системы Telegram-бота о рыночных активах

# **Разработка алгоритма решения задачи**

* 1. **Алгоритм взаимодействия с пользователем**

Описание:

1. Пользователь отправляет запрос (команду или нажатие кнопки) через Telegram-бот.
2. Бот определяет тип запроса и вызывает соответствующий обработчик.
3. Обработчик выполняет действия:
   * Извлечение данных из базы данных или внешнего API.
   * Обновление настроек пользователя при необходимости.
   * Формирование ответа для пользователя.
4. Ответ отправляется пользователю через бот.

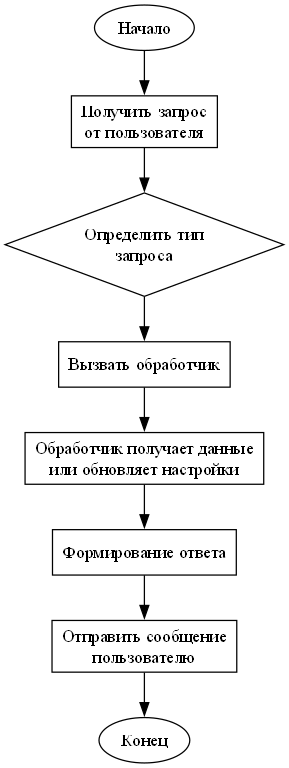


Рисунок 19 – Алгоритм взаимодействия с пользователем

* 1. **Алгоритм регистрации пользователя**

Описание:

1. Получить идентификатор пользователя (user\_id).
2. Проверить наличие записи пользователя в таблице User.
3. Если запись отсутствует:
   * Создать новую запись в таблице User.
   * Создать начальные настройки для пользователя в таблице Settings.
4. Завершить процесс.

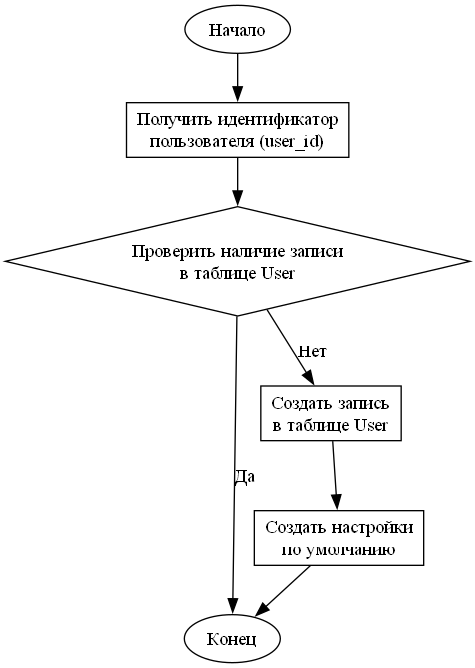


Рисунок 20 – Алгоритм регистрации пользователя

* 1. **Алгоритм получения данных о задаче**

Описание:

1. Получить запрос от пользователя на получение данных.
2. Определить тип данных, которые запрашиваются.
3. Извлечь данные из базы данных или внешнего API.
4. Обработать и форматировать данные.
5. Отправить данные пользователю.

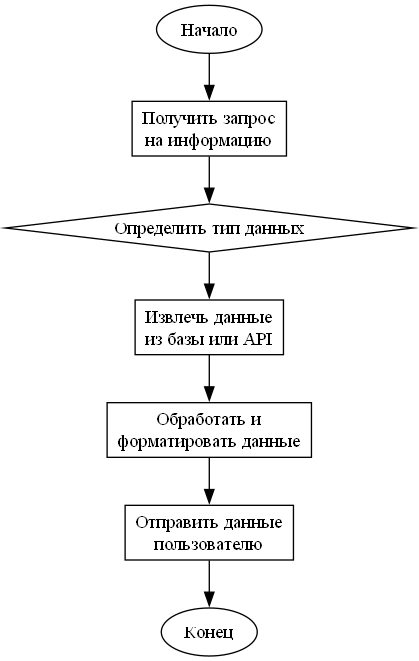


Рисунок 21 – Алгоритм получения данных о задаче

* 1. **Алгоритм настройки уведомлений**

Описание:

1. Получить запрос на настройку уведомлений.
2. Определить параметры уведомлений (тип, значение и условия).
3. Сохранить настройки в базе данных (в таблице Settings).
4. Отправить подтверждение пользователю.

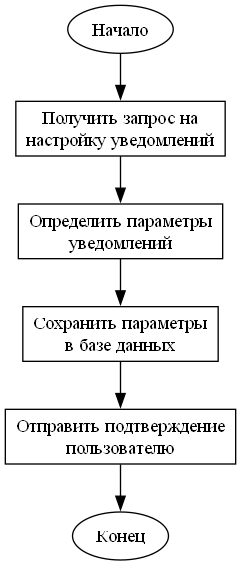


Рисунок 22 – Алгоритм настройки уведомлений

* 1. **Алгоритм отправки уведомлений**

Описание:

1. Периодически запускать задачу планировщиком (например, раз в 10 минут).
2. Для каждого пользователя извлечь настройки уведомлений.
3. Проверить текущие данные и условия для отправки уведомлений.
4. Если условия выполнены:
   * Сформировать уведомление.
   * Отправить его пользователю.
   * Обновить статус уведомления в базе данных.

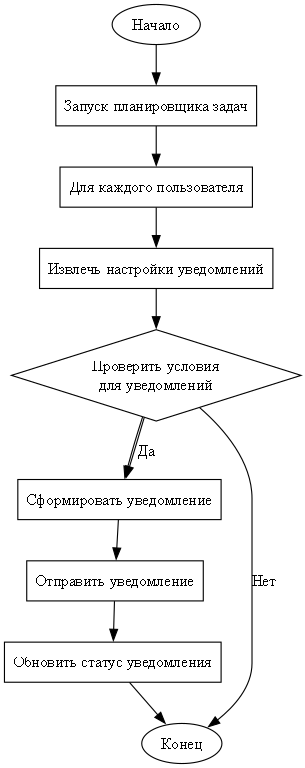


Рисунок 23 – Алгоритм отправки уведомлений

# **Выбранная архитектура системы**

## **6.1. Общая архитектура**

* Система построена на основе клиент-серверной архитектуры с использованием асинхронных технологий для обеспечения высокой производительности и масштабируемости. Архитектура системы состоит из следующих компонентов:
* Клиентская часть:  
  Интерфейс пользователя реализован через Telegram-бот, который предоставляет доступ к функциональности системы.
* Серверная часть:  
  Серверная логика реализована на Python с использованием FastAPI для REST API и Aiogram для взаимодействия с пользователями через Telegram.
* База данных:  
  Хранит информацию о пользователях, их настройках и избранных активах. Для небольших нагрузок используется SQLite, для более масштабных задач — PostgreSQL.
* Внешние API:  
  Интеграция с внешними сервисами для получения данных об активах (например, Yahoo Finance, Investpy, Tinkoff).

## **6.2. Технологический стек**

* Язык программирования: Python 3.9+.
* Фреймворки и библиотеки:
  + Aiogram: Асинхронный фреймворк для Telegram-ботов.
  + FastAPI: Для реализации REST API.
  + SQLAlchemy Async: Для асинхронного взаимодействия с базой данных.
  + Asyncio: Для реализации асинхронных задач.
  + Httpx: Асинхронный HTTP-клиент для работы с внешними API.
* База данных: SQLite для локального хранения данных или PostgreSQL для более сложных сценариев.
* Среда выполнения: Сервер или облачный сервис с поддержкой Python и необходимых библиотек.

## **6.3. Структура программы**

* Программа имеет четко структурированную организацию, основанную на принципах модульности и разделения обязанностей:
* Корневой каталог:
  + Документация проекта: техническое задание, руководство программиста, аналитическая записка и т.д.
  + Файл requirements.txt для управления зависимостями.
* Модуль app:
  + main.py: Основной файл для запуска серверного приложения.
  + Подкаталоги:
    - src/endpoints: Эндпоинты FastAPI для обработки запросов (получение информации об активах, графиков и т.д.).
    - src/services: Модули для работы с внешними API (Investpy, Yahoo Finance, Tinkoff).
    - src/\_\_init\_\_.py: Инициализация структуры проекта.
* Модуль bots:
  + main.py: Файл для запуска Telegram-бота.
  + Подкаталоги:
    - handlers: Логика обработки команд, состояний и сообщений.
    - keyboards: Генерация клавиатур для взаимодействия с пользователем.
    - data: Скрипты для взаимодействия с базой данных.
    - other: Утилиты для построения графиков, отправки уведомлений и работы с данными.
    - utils: Вспомогательные модули, такие как обработчики ошибок и middlewares.