МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет	*Кафедра*
вычислительной техники	«Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»

Профиль подготовки: «Программное обеспечение средств вычислительной техники и автоматизированных систем»

БАКАЛАВРСКАЯ РАБОТА

на тему

ВК-БОТ

«НАПОМИНАНИЯ»

Студент Дасаев 17.06.2026 Дасаев Тимур Зягитович
(подпись, дата) (ФИО полностью)

Руководитель 17.06.2026 Н.В. Слепцов
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр 17.06.2026 А.С. Бычков
(подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 22.05.2026 г. № 14)

Заведующий кафедрой _____ М.А. Митрохин
(подпись) (фамилия, инициалы)

Работа защищена с отметкой удовлетворительно (протокол заседания ГЭК от 19 июня 2026 г. № 43)

Секретарь ГЭК _____ Е.И. Гурин
(подпись) (фамилия, инициалы)

Пенза 2026

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

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
_____М.А. Митрохин
«07» 04 2026 г.

ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ

1. Студент. Дасаев Тимур Зягитович. гр. 22ВВП1 факультета вычислительной техники

2.Тема работы «ВК-бот «Напоминания»»

Тема утверждена приказом ПГУ №347/0 от "07" 04 2026 г.

3. Руководитель бакалаврской работы к.т.н. доцент кафедры ВТ Слепцов Н.В.

4. Задание на работу (назначение разработки, исходные данные и т.п.)
Необходимо спроектировать и разработать VK-бота, который будет управлять системой напоминаний с расширенным функционалом. Бот должен обеспечивать: создание обычных напоминаний с указанием даты и времени, создание напоминаний с предварительным уведомлением (за несколько дней до события), создание повторяющихся напоминаний (каждый час, каждый день, каждую неделю), гибкие настройки времени и периодичности напоминаний, удобный интерфейс с использованием кнопок для просмотра, управления и удаления напоминаний, надежное хранение данных в базе данных, точную и своевременную отправку уведомлений пользователю.

5. Перечень подлежащих разработке вопросов

Требуется произвести анализ предметной области: необходимо изучить возможные альтернативы данному инструменту, другие площадки, на которых возможно базирование VK-бота, а также способы разработки бота, в частности создание на базе Visual Studio 2022 и языка программирования C#.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области	27	07.04.2026-08.04.2026	
2	Анализ аналогов	22	09.04.2026-14.04.2026	
3	Постановка задачи	25	15.04.2026-18.04.2026	
4	Выбор инструментов разработки	30	19.04.2026-25.04.2026	
5	Настройка инструментов разработки	28	26.04.2026-01.05.2026	
6	Разработка архитектуры базы данных	45	02.05.2026-08.05.2026	
7	Разработка ВК-бота	55	8.05.2026-13.05.2026	
8	Тестирование бота	55	13.05.2026-18.05.2026	
9	Оформление пояснительной записки	39	18.05.2026-21.05.2026	

Дата выдачи задания "07" 04 2026 г.

Руководитель бакалаврской работы _____ Слепцов Н.В.
(подпись, дата)

Задание к исполнению принял студент _____ Дасаев Т.З.
(подпись, дата)

Работу к защите допустить
Декан факультета _____ 17.06.2026 Фионова Л.Р.
(подпись, дата)

Реферат

Бакалаврская работа содержит 108 листов, 30 рисунков, 3 таблицы, 13 источников, 3 приложения.

VK-БОТ, ДОБАВЛЕНИЕ НАПОМИНАНИЙ, НАПОМИНАНИЯ, БАЗА ДАННЫХ, C#, VISUAL STUDIO, SQLITE, QUARTZ

Цель работы – разработать VK-бота, который будет управлять напоминаниями с расширенным функционалом, позволяющим упростить повседневную жизнь пользователя.

Одним из путей достижения поставленной задачи является VK-бот, обеспечивающий выполнение следующих функций: создание обычных, с предварительным уведомлением и повторяющихся напоминаний, гибкие настройки времени и периодичности, удобный интерфейс для просмотра и удаления напоминаний, надежное хранение данных и точную отправку уведомлений.

В конечном итоге разработано приложение, полностью соответствующее цели данной выпускной квалификационной работы. В ходе достижения результата были использованы следующие технологии: Visual Studio 2022, C#, SQLite, Quartz.NET. Приложение протестировано и готово к эксплуатации.

Содержание

Перечень сокращений и обозначений …… 5
Введение…….6
1 Telegram-бот «Напоминания»…….. 7
1.1 Анализ предметной области ……..7
1.2 Обзор аналогов …….8
1.3 Постановка задачи …….. 14
1.3.1 Назначение бота ……14
1.3.2 Функциональные требования ……. 14
1.3.3 Команды бота ……..... 15
1.3.4 Технические требования …… 15
1.3.5 Дополнительные возможности ……..... 15
1.3.6 Критерии успешной реализации ……..... 15
1.3.7 Минимальные системные требования ……... 16
1.4 Архитектура приложения ……..... 16
1.4.1 Общая структура ……...……. 16
1.4.2 Основные компоненты …….. 17
1.4.3 Взаимодействие компонентов ……... 17
1.5 Выбор инструментов разработки ……...……..19
1.5.1 Язык программирования …….……19
1.5.2 Библиотеки и фреймворки …….....19
1.5.3 Среда разработки …… 20
1.5.4 Настройка среды разработки VisualStudio 2022 ……..... 21
1.5.5 Настройка и подключение базы данных …….……..... 34
2 Проектирование и разработка ……..…….... 35
2.1 Основные компоненты ……..……... 37
2.1.1 BotBackgroundService ……. 30
2.1.2 ReminderService …….…….. 32

2.1.3 DatabaseService ……..……... 33
2.1.4 ReminderJob …….…..... 35
2.2 Тестирование ……. 36
3 Безопасность жизнедеятельности …….…… 46
3.1 Правила безопасности при использовании ПК …….…..... 46
3.2 Правила безопасности при использовании смартфона ……...…. 46
Заключение …….…… 50
Список использованных источников …….…….. 51
Приложение А –UML-диаграммы…….……... 52
Приложение Б – Листинг программы …….…… 57
Приложение В – Презентация …….…… 82

Перечень сокращений и обозначений

БД – база данных

ОС – операционная система

ПК – персональный компьютер

API (ApplicationProgrammingInterface) – программныйинтерфейсприложения

DI (DependencyInjection) – паттерн проектирования, который позволяет объектам программы внедрять зависимости

Git (GlobalInformationTracker) – распределенная система управления версиями

GSM (Groupe Special Mobile) – стандартсотовойсвязи

ID (identifier) – идентификатор

IDE (IntegratedDevelopmentEnvironment) – программное обеспечение, которое помогает разработчикам писать и тестировать код

IOS (IphoneOperatingSystem) – мобильная операционнаясистема

LTE (Long Term Evolution) – долговременное развитие относящееся к технологии 4G

MacOS (MacintoshOperatingSystem) – операционнаясистемаперсонального компьютера

ORM (Object-RelationalMapping) – объектно-реляционное отображение

SDK (SoftwareDevelopmentKit) – пакет инструментов, предназначенный для разработки программного обеспечения

SQL (StructuredQueryLanguage) – язык структурированных запросов

UML (UnifiedModelingLanguage) – графический язык, используемый для визуализации, описания и документирования программных систем и бизнес-процессов

URL (Uniform Resource Locator) – адресресурса в интернете

VS (VisualStudio) – линейка систем разработки программного обеспечения от компании Microsoft

Введение

В современном быстро меняющемся мире, характеризующемся постоянным ростом объема задач и обязанностей, эффективное управление временем приобрело особую важность. Одной из распространенных проблем является забывчивость, из-за которой пользователи пропускают важные события, сроки выполнения задач и запланированные встречи. Традиционные способы организации напоминаний, такие как бумажные заметки или стандартные календарные приложения, не всегда обеспечивают достаточный уровень удобства и эффективности.

Одним из эффективных решений этой проблемы являются боты, работающие в социальной сети «ВКонтакте». Они позволяют пользователям создавать, редактировать и получать напоминания непосредственно в интерфейсе мессенджера. Благодаря широкой аудитории платформы и ее кроссплатформенной совместимости такие инструменты используются как в повседневной жизни, так и на рабочем месте.

Целью данной работы является разработка бота для «ВКонтакте», предназначенного для управления системой управления временем пользователя и создания систем оповещения(напоминания), предназначенный для планирования рабочего и личного времени. Предполагается как оснащение разрабатываемого бота расширенными возможностями, так и развитие системы возможностей в соответствии с требованиями пользователей. К расширенным возможностям в частности относится:

– введение стандартных списков наиболее часто планируемых мероприятий, в том числе с разной временной периодичностью.

– создание различных типов напоминаний (стандартных, с предварительным уведомлением и повторяющихся);

– настройка параметров времени и частоты с высокой степенью гибкости;

– реализация удобного и практичного интерфейса для просмотра, редактирования и удаления напоминаний;

– обеспечение безопасного хранения данных и своевременной доставки уведомлений.

1 ВК-бот «Напоминания»

1.1 Анализ предметной области

В последние годы взаимодействие с ботами в мессенджерах постепенно вытеснило традиционные методы ведения списков дел и ручной настройки напоминаний. Все больше пользователей предпочитают поручать планирование своих повседневных задач автоматизированным системам, а не пользоваться записными книжками или полагаться на свою память.

В нынешнее время на выбор социальной сети для создания чат-бота дается лишь несколько вариантов. В приоритете сервисы, которые в белых списках, чтобы пользователь мог пользоваться чат-ботом в любое удобное время. На данный момент таких сервисов три: мессенджеры «Макс», «ВКонтакте» и «Одноклассники».

Существенным недостатком в мессенджере «Макс» создавать чат-боты могут только юридические лица. Для того чтобы активировать бота в «Максе», нужно перейти в приложение «Макс для бизнеса» и авторизоваться. Затем необходимо создать профиль и пройти верификацию. В разделе «чат-бот и мини приложение» нужно нажать «Создать», заполнить карточку и отправить на модерацию.

В мессенджере «Одноклассники» создать чат-бота можно, но реализовать сложнее. У «Одноклассников» есть API для ботов, но он значительно проще и ограниченнее чем в других популярных мессенджерах. В нем меньше примеров кода, меньше готовых библиотек, а документация менее полная. Реализация сложной логики (такой как несколько типов

напоминаний, клавиатуры и восстановление после перезапуска) будет заметно сложнее и займет больше времени.

Тем временем «ВК» предлагает хорошо документированный и мощный API, множество готовых библиотек, простую интеграцию с базами данных и широкий выбор элементов визуального интерфейса.

Для реализации наиболее предпочтительной платформы среди социальных сетей был проведен сравнительный анализ на величину аудитории в каждой социальной сети. Данные по средней месячной аудитории представлены в таблице 1.

Таблица 1 – Сравнение месячной аудитории Вонтакте, Одноклассников и Макса в России

Социальная сеть	Средняя месячная аудитория	Период
Вконтакте	92-93 млн	Конец 2025
Одноклассники	34 млн	Конец 2025
Макс	50-55 млн	Конец 2025 – начало 2026

Из данных таблицы следует, что из всех возможных вариантов социальная сеть «ВКонтакте» является наиболее подходящей для разработки чат-бота из-за большей аудитории, а также и функциональных возможностей. Современные боты в «ВКонтакте» предлагают расширенные возможности: помимо отправки уведомлений в заданное время, они могут анализировать повторяющиеся действия, интегрироваться с календарными сервисами и создавать более эффективные расписания на основе привычек пользователя.

Как правило, такие системы хранят информацию обо всех запланированных пользователем событиях – от разовых задач до регулярно повторяющихся действий. На основе этих данных можно сделать выводы о

рабочей нагрузке человека, его повседневных привычках (например, когда он начинает свой день или делает перерывы), а также определить важные даты, включая дни рождения и сроки выполнения задач.

Разработчики ботов для VK [1], от крупных организаций до небольших команд, в основном используют собранные данные следующим образом:

– улучшение функциональности, например, путем внедрения шаблонов напоминаний или интеллектуальных функций планирования;

– персонализация пользовательского опыта путем адаптации интерфейса и системы уведомлений к индивидуальным предпочтениям пользователя.

Следует отметить, что не только разработчики заинтересованы в таких данных. Различные категории пользователей также могут эффективно применять возможности ботов в своей работе. Например, в корпоративном секторе их используют для распределения задач и координации рабочих процессов внутри команд.

В сфере образования такие инструменты помогают своевременно информировать студентов о расписании занятий, предстоящих лекциях и сроках сдачи заданий. Медицинские учреждения, в свою очередь, используют ботов для напоминания пациентам о приеме лекарств и посещении приёмов у специалистов.

Анализ статистики использования ботов предоставляет дополнительные возможности для выявления закономерностей в поведении пользователей. Это позволяет определять пиковые периоды для планирования задач, оценивать уровень загруженности и делать прогнозы относительно будущих действий. Например, система может корректировать время отправки уведомлений, предлагая вечерние напоминания пользователям, которые обычно планируют свои задачи на утро.

1.2 Обзор аналогов

Изучив множество источников сети интернет, выявилось, что приложения с напоминаниями имеют большую популярность. К сожалению, про чат-ботов , такое не скажешь. Так как пользователи привыкли к удобству в разных приложениях. Это говорит о том, что в ботах нет такого функционала, как в популярных приложениях с напоминаниями.

Первым на очереди «Напоминалка» – бот для простого и удобного управления задачами.

В нем создаются напоминания на нужные даты и время, планируются дела и автоматически приходят уведомления о важных делах и задачах прямо в VK. Данный бот представлен на рисунках 1, 2.

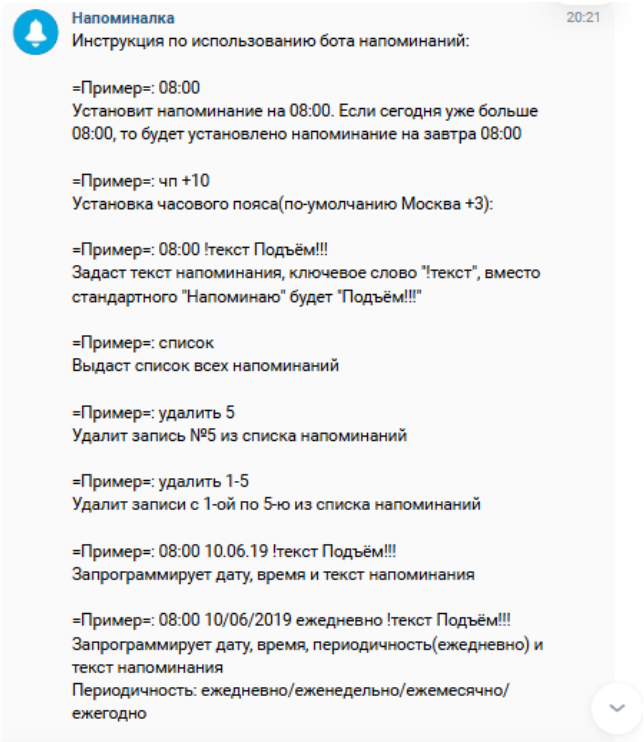


Рисунок 1 – Начальное окно приветствия

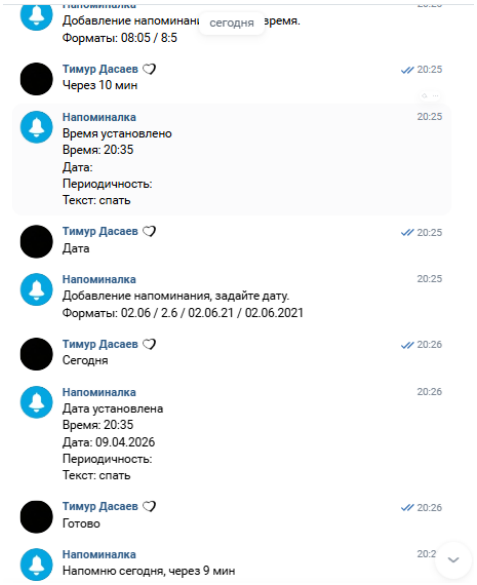


Рисунок 2 – Создание и получение напоминания

Вторым по счету бот «Нотификейк» — это удобный бот внутри VK, который помогает не забывать о важных делах, встречах и привычках. Он работает как персональный секретарь, позволяя быстро создавать напоминания без установки дополнительных приложений. Функционал схож с телеграм-ботом, что представлен ранее, но есть некоторые преимущества и недостатки, ознакомиться можно на рисунках 3-6.

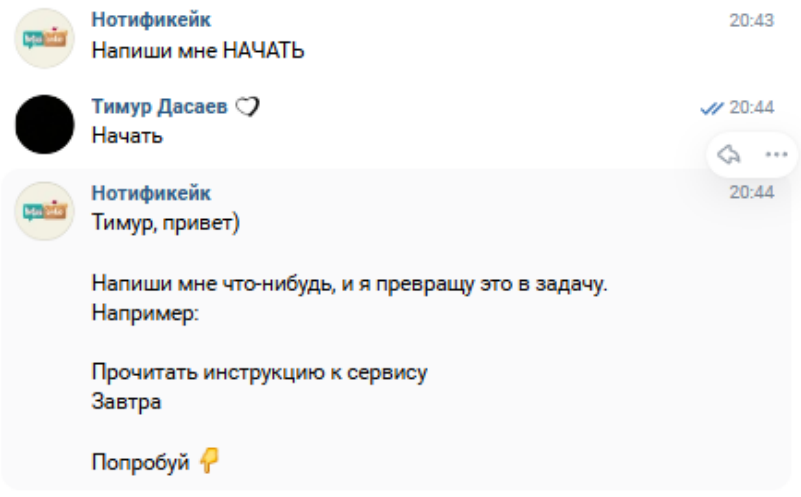


Рисунок 3 – Начальное окно приветствия и настройки бота

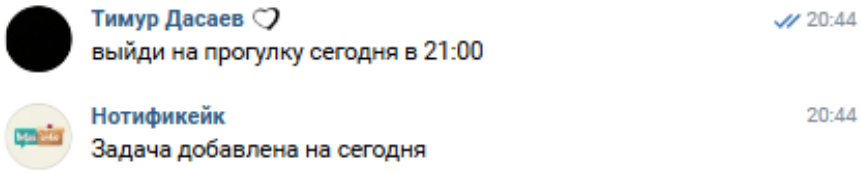


Рисунок 4 – Создание напоминания

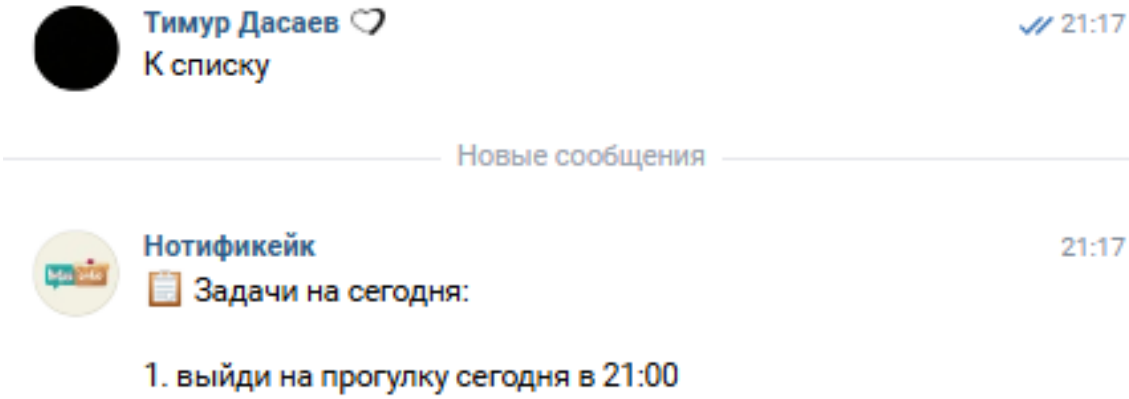


Рисунок 5 – Список созданных напоминаний

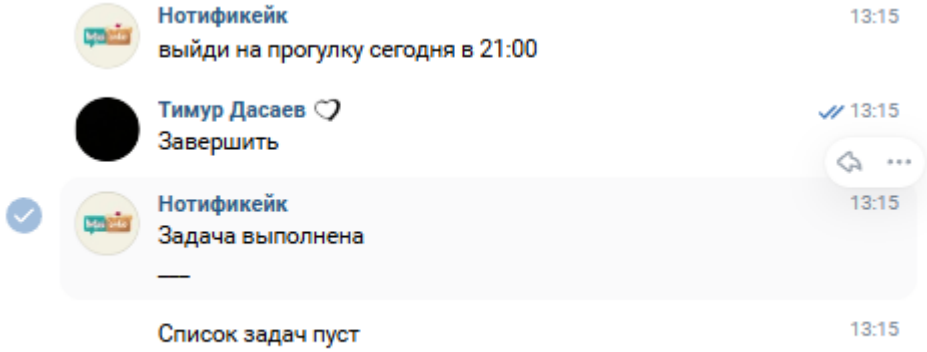


Рисунок 6 – Завершение задачи

Ознакомившись с ботами, следующий выбор пал на популярное приложение, такое как, Google Keep [2] – это простой сервис для создания заметок и напоминаний от Google. В который можно добавлять напоминания к любой заметке, и они будут полностью синхронизированы по воздуху на всех устройствах. Скриншоты сервиса представлены ниже на рисунке 7.

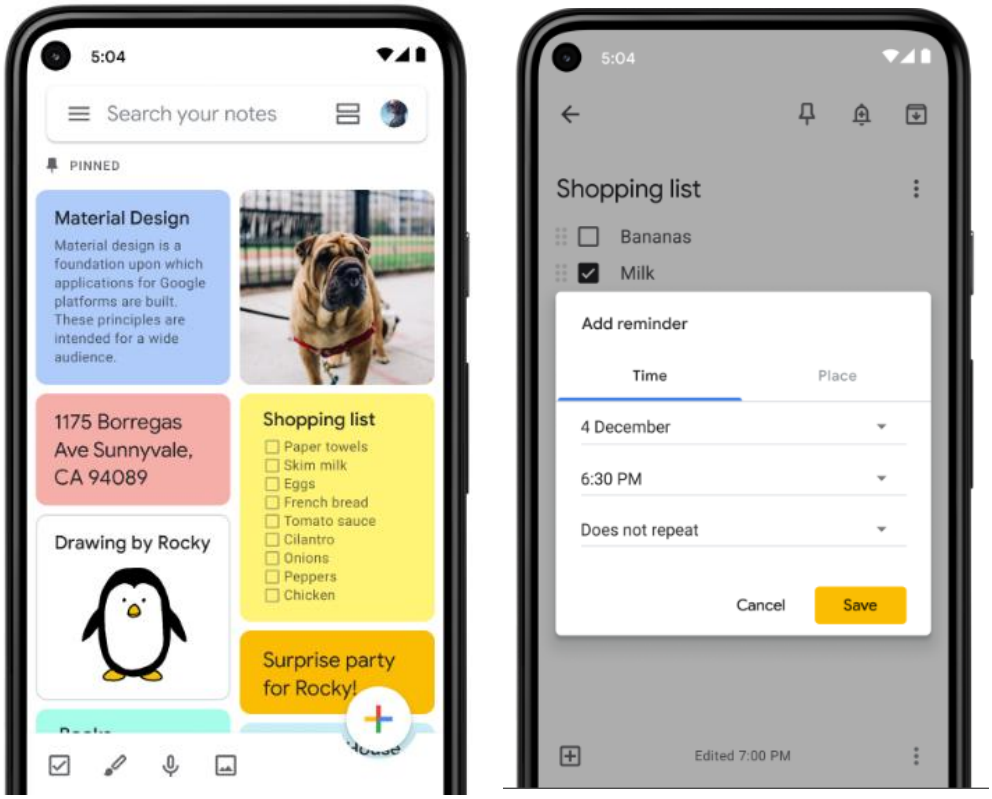


Рисунок 7 – Интерфейс GoogleKeep

Следующий выбор пал на популярное приложение, такое как, Todoist [3] – это современное приложение для управления задачами и ежедневного планирования, которое позволяет пользователям эффективно организовывать свою работу и личные дела. Сервис дает возможность создавать задачи и подзадачи, устанавливать сроки выполнения, настраивать напоминания (в том числе повторяющиеся) и группировать задачи по проектам и приоритетам. Благодаря интуитивно понятному интерфейсу и поддержке различных платформ Todoist широко используется как для индивидуального планирования, так и для работы в команде. Дополнительные функции, такие как аналитика продуктивности и интеграция с другими сервисами, делают его универсальным инструментом для повышения эффективности и отслеживания выполнения задач.

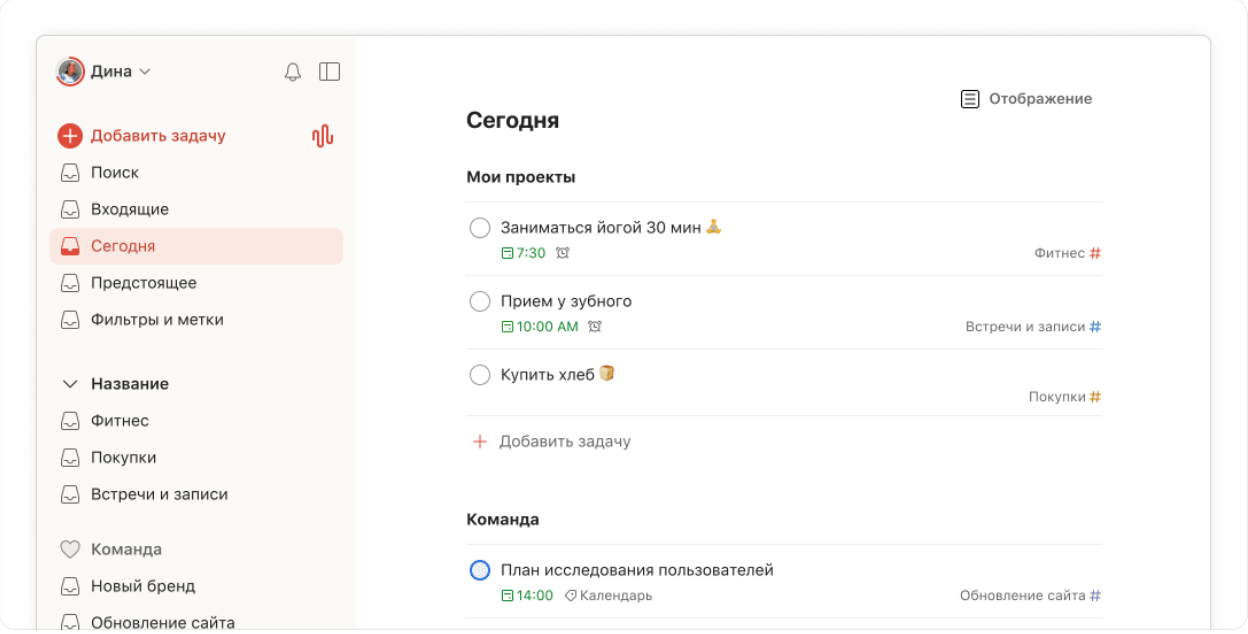


Рисунок 8 – Интерфейс Todoist

Не менее популярным является приложение TickTick[4]– это современное приложение для управления задачами и планирования времени, сочетающее в себе функции списка дел и календаря. Сервис позволяет создавать задачи и подзадачи, настраивать напоминания (в том числе повторяющиеся и с несколькими уведомлениями), а также гибко настраивать свой график с помощью различных режимов просмотра, таких как список, календарь или доска Канбан. Кроме того, приложение поддерживает синхронизацию между устройствами, интеграцию с внешними сервисами и совместную работу, что расширяет его возможности как для личного, так и для командного использования.

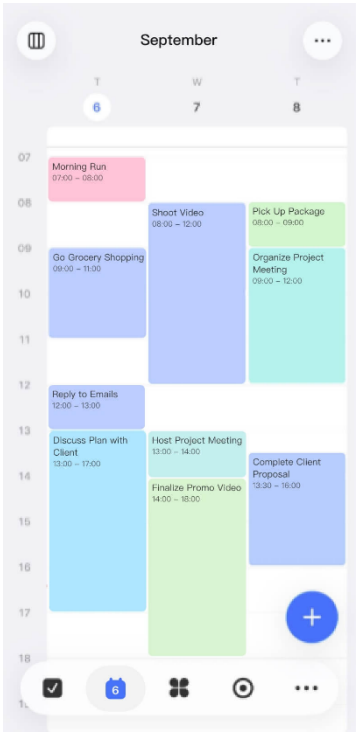


Рисунок 9 – Интерфейс MicrosoftToDo

После анализа и тестирования различных сервисов для управления напоминаниями, включая Google Keep, Todoist, TickTick и чат-ботов ВКонтакте, были сделаны следующие выводы.

Google Keep зарекомендовал себя как удобный инструмент для создания быстрых личных заметок и напоминаний благодаря простому интерфейсу и тесной интеграции с другими продуктами Google. Однако его функционала недостаточно для организации сложных задач и многоуровневого планирования.

Todoist, с другой стороны, представляет собой более продвинутую систему управления задачами, позволяющую пользователям эффективно структурировать проекты и отслеживать выполнение задач. Однако использование этого сервиса требует регистрации, а некоторые функции могут быть менее удобны при работе с контекстными или геолокационными напоминаниями.

В отличие от предыдущих решений, TickTick сочетает в себе функционал планировщика с инструментами для повышения продуктивности. Он поддерживает гибкие настройки напоминаний, различные форматы отображения задач и дополнительные функции, такие как таймер «Помодоро» и трекер привычек, что делает его универсальным инструментом как для личного, так и для профессионального использования.

Анализ ботов ВК показал, что такие решения, как «Нотификейк» и «Напоминалка», являются эффективными вариантами для реализации простых и гибких напоминаний без необходимости установки дополнительных приложений. Их основными преимуществами являются прямой доступ в мессенджере и удобный интерфейс кнопок. Однако по сравнению с полнофункциональными сервисами их функционал остается ограниченным, особенно в плане интеграции с внешними системами и расширенного управления задачами.

В итоге можно сказать, что классические мобильные приложения для напоминаний (такие как Todoist, TickTick, Google Keep и Microsoft To Do) имеют несколько значительных преимуществ по сравнению с чат-ботами. Во-первых, эти приложения предлагают удобный графический интерфейс с поддержкой календаря, различными видами отображения задач, подзадачами, приоритетами и аналитикой продуктивности, что позволяет пользователям эффективно управлять большими объёмами информации. Во-вторых, приложения более надежны в отправке push-уведомлений, в работе в фоновом режиме. Также полноценные приложения предоставляют глубокую интеграцию системой и, что немаловажно, могут работать в офлайн режиме. В отличие от чат-ботов, приложение не зависит от мессенджера, предлагают более расширенные возможности. Например: трекер привычек, напоминания с геолокацией и с синхронизацией с другими устройствами пользователя. Из-за этих всех преимуществ мобильные приложения остаются более

популярным, более практичным и удобным для управлениями напоминаниями.

Таким образом, можно сделать вывод, что существующие решения имеют различные преимущества и ограничения, а выбор конкретного инструмента зависит от задач пользователя: от простых напоминаний до комплексного управления временем и проектами.

Таблица 2 – Преимущества и недостатки телеграм-ботов и дополнительных сервисов для напоминаний

Задачи	Напоминалка	Нотифитейк	Todoist	Google Keep	TickTick
Простота использования	+	+	+	+	+
Синхронизация с календарями	-	-	+	+	+
Поддержка нескольких языков	-	-	+	+	+
Создание голосовых напоминаний	-	-	-	+	-
Гибкие настройки	-	-	+	-	+
Интеграция с внешними сервисами	-	-	+	+	+
Веб-версия	+	+	+	+	+
Повторяющиеся задачи	+	+	+	+	+
Подходит для командной работы	-	-	+	+	+
Кроссплатформенность	Vk mobile + Web	Vk mobile + Web	Все платформы	Все платформы	Все платформы

1.3 Постановка задачи

Постановка задачи для Vk чат-бота "Бот напоминаний" (vk.com/bot_napominaniy) была следующей:

1.3.1 Назначение бота

Vk-бот предназначен для создания, управления и отправки напоминаний пользователям в указанное время. В отличие от аналогов, имеются разделения на добавление напоминаний.

1.3.2 Функциональные требования

Функциональные возможности системы предусматривают создание напоминаний с использованием кнопки «Добавить напоминание». Поддерживаются следующие типы напоминаний:

– обычное напоминание, отправляемое в заданную дату и время;

– напоминание с предварительным уведомлением, при котором сообщение отправляется за два дня и за один день до наступления события;

– повторяющееся напоминание с возможностью выбора периода: «каждый час», «каждый день», «каждую неделю».

1.3.3 Команды бота

В случае недоступности графического интерфейса (кнопок управления) предусмотрено взаимодействие с ботом посредством текстовых команд. Основные команды представлены ниже:

– /start — запуск бота и отображение главного меню;

– /add — создание нового напоминания;

– /list — вывод списка активных напоминаний;

– /delete [ID] — удаление напоминания по идентификатору;

– /help — отображение справочной информации по использованию бота.

1.3.4 Технические требования

Разработка VK-бота выполнена с использованием языка программирования C#. В качестве среды разработки использовалась Visual Studio 2022.

В процессе реализации применялись следующие библиотеки и технологии:

– библиотека для взаимодействия с API социальной сети ВКонтакте;

– Microsoft.EntityFrameworkCore — для работы с базой данных.

Для планирования и обработки задач использовался Quartz.NET в связке с базой данных SQLite, обеспечивающий хранение данных и выполнение отложенных операций по расписанию.

1.3.5 Дополнительные возможности

Интерфейс взаимодействия с пользователем реализован с использованием следующих элементов:

– ReplyKeyboardMarkup (кнопки быстрого доступа в интерфейсе диалога);

– InlineKeyboardMarkup (встроенные кнопки в сообщениях бота).

1.3.6 Критерии успешной реализации

Система считается успешно реализованной при выполнении следующих условий:

– стабильная работа VK-бота без критических ошибок;

– корректная и своевременная доставка уведомлений;

– наличие удобного и интуитивно понятного пользовательского интерфейса;

– отсутствие потери или искажения данных при обработке напоминаний.

1.3.7 Минимальные системные требования

Требования для пользователей:

– устройство: смартфон, планшет или персональный компьютер с доступом к ВКонтакте;

– операционная система: Android 7.0 и выше, iOS 13 и выше, Windows 10 и выше, macOS 10.15 и выше;

– наличие стабильного интернет-соединения (3G/LTE или Wi-Fi);

– установленное приложение ВКонтакте или доступ через веб-версию.

Требования для разработки:

– среда разработки: Visual Studio 2022 (или аналогичная IDE);

– платформа разработки: .NET 6.0 и выше;

– поддержка C# 10.0 и выше.

На рисунке 10 представлена UML-диаграмма вариантов использования разработанного VK-бота.

Рисунок 10 – UML-диаграмма использования

1.4 Архитектура приложения

1.4.1 Общая структура

Разрабатываемое приложение построено на основе многослойной архитектуры, обеспечивающей разделение ответственности между основными компонентами системы. Выделяются следующие уровни:

– уровень взаимодействия с пользователем, включающий обработку входящих сообщений и команд, а также отправку уведомлений через социальную сеть ВКонтакте;

– уровень бизнес-логики, реализующий управление напоминаниями, включая их создание, обработку и удаление;

– уровень доступа к данным, обеспечивающий взаимодействие с базой данных SQLite и хранение информации о напоминаниях;

– уровень планирования задач, реализованный с использованием библиотеки Quartz.NET и предназначенный для выполнения отложенных операций.

Такое разделение позволяет повысить модульность системы, упростить её сопровождение и обеспечить гибкость при дальнейшем расширении функционала.

1.4.2 Основные компоненты

Ключевые компоненты системы, обеспечивающие корректную и стабильную работу VK-бота, представлены в таблице 2.

Таблица 3 – Основные компоненты системы

Компоненты	Назначение	Технологии
VkBotPollingService	Фоновый сервис для получения и обработки входящих сообщений пользователей	VkNet, .NET Hosting

NotificationManagerService	Реализация бизнес-логики управления напоминаниями (обработка, создание, удаление)	C#, Quartz.NET
NotificationStorageService	Обеспечение взаимодействия с базой данных и хранение напоминаний	SQLite
NotificationJob	Задача планировщика для отправки уведомлений в заданное время	Quartz.NET

1.4.3 Взаимодействие компонентов

Взаимодействие компонентов системы осуществляется по последовательной схеме. Пользователь отправляет сообщение боту в социальной сети ВКонтакте, после чего оно поступает в фоновый сервис VkBotPollingService, выполняющий его первичную обработку и анализ.

Далее запрос передаётся в компонент бизнес-логики — NotificationManagerService, который отвечает за обработку пользовательских данных, формирование напоминания и управление его жизненным циклом. В процессе работы сервис сохраняет информацию о напоминании с помощью NotificationStorageService, а также регистрирует задачу в системе планирования.

Планирование и выполнение отложенных задач осуществляется с использованием библиотеки Quartz.NET. В заданное время планировщик инициирует выполнение задачи NotificationJob, которая, в свою очередь, отправляет уведомление пользователю через сервис отправки сообщений.

Таким образом, взаимодействие компонентов реализуется по следующей цепочке: пользователь → сервис обработки сообщений → сервис бизнес-логики → система хранения данных и планировщик задач. При

наступлении заданного времени происходит обратный процесс: планировщик → задача → отправка сообщения пользователю.

UML-диаграмма взаимодействия компонентов представлена на рисунке 11.

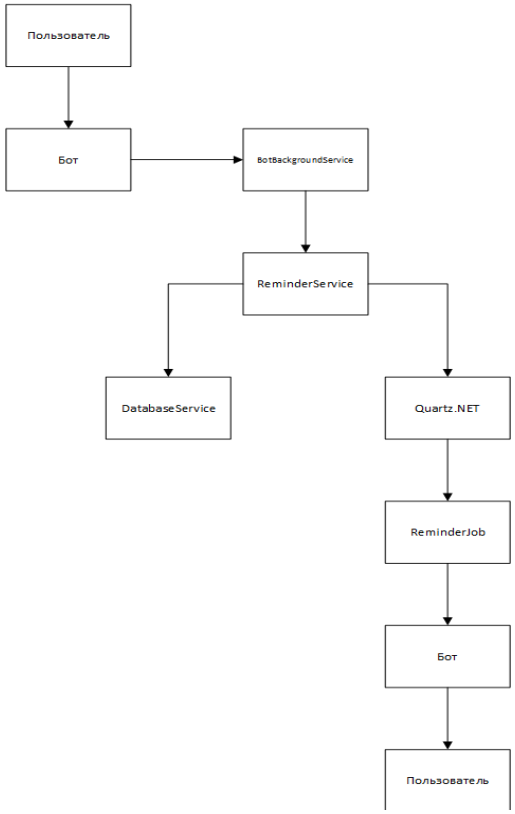


Рисунок 11 – UML-диаграмма взаимодействия компонентов

1.5 Выбор инструментов разработки

1.5.1. Язык программирования

C# (.NET 8.0)[4]– этот язык программирования отличается высокой производительностью, полной поддержкой асинхронного программирования, а также наличием обширной экосистемы библиотек.

1.5.2. Библиотеки и фреймворки

Разрабатываемый в настоящее время бот для «ВКонтакте», предназначенный для управления уведомлениями, использует следующие ключевые библиотеки, каждая из которых играет важную роль в архитектуре приложения:

VkNet– библиотека-обёртка для APIVK (версия 5.x), предоставляющая удобный интерфейс для взаимодействия с функциями социальной сети «ВКонтакте». Она отвечает за:

– отправку и получение сообщений с помощью метода messages.send;

– работу с клавиатурами (MessageKeyboard);

– взаимодействие с сообществами и пользователями;

– упрощение вызовов APIVK без необходимости прямых HTTP-запросов.

Библиотека значительно упрощает интеграцию приложения с VK, предоставляя готовые методы для работы с сообщениями и интерфейсом бота. Quartz.NET [5] — мощный планировщик задач, используемый для:

– выполнения напоминаний точно в указанное пользователем время;

– поддержки повторяющихся задач (ежедневные, еженедельные напоминания);

– обеспечения надежности — задачи сохраняются при перезапуске приложения.

Quartz.NET позволяет гибко планировать задачи с помощью триггеров и гарантирует своевременное их выполнение.

Microsoft.Data.Sqlite [6] – это облегченный провайдер ADO.NET для работы с SQLite, который:

– обеспечивает взаимодействие с локальной файловой базой данных;

– позволяет выполнять SQL-запросы для хранения и извлечения данных напоминаний;

– поддерживает транзакции для обеспечения целостности данных.

Эта библиотека была выбрана за ее простоту и эффективность для небольших проектов.

EntityFrameworkCore — это объектно-реляционный маппер (ORM), который:

– позволяет работать с базой данных через объекты C#;

– автоматизирует создание и миграцию схемы базы данных;

– предоставляет интерфейс LINQ для написания типобезопасных запросов.

EntityFramework значительно ускоряет разработку, устраняя необходимость писать сырые SQL-запросы.

Каждая из этих библиотек была выбрана исходя из требований проекта: необходимости надежного взаимодействия с APIVK, точного планирования задач, простого локального хранения данных и удобного доступа к этим данным. Вместе они образуют мощную, но легкую основу для Вконтакте-бота, обеспечивая всю необходимую функциональность без излишнего усложнения архитектуры.

1.5.3 Среда разработки

Бот для VK был разработан с использованием MicrosoftVisualStudio 2022 – современной среды разработки и профессиональной интегрированной среды разработки (IDE) для платформы .NET, которая предоставляет полный набор инструментов для эффективной разработки [7].

MicrosoftVisualStudio 2022 был выбран по следующим причинам:

Во-первых, среда обеспечивает полную поддержку .NET 8.0, гарантируя совместимость с современными возможностями языка C#, включая повышенную производительность и поддержку асинхронных операций.

Во-вторых, VisualStudio имеет глубокую интеграцию с системой контроля версий Git, что упрощает управление кодом и командную разработку.

Одним из ключевых преимуществ является мощный встроенный отладчик, который позволяет:

– устанавливать точки останова;

– выполнять пошаговую отладку;

– отслеживать значения переменных в режиме реального времени;

– анализировать поведение приложения во время выполнения.

Также используется встроенный менеджер NuGet, который обеспечивает удобную установку и обновление библиотек (VkNet, Quartz.NET и др.) без необходимости вручную редактировать конфигурационные файлы

1.5.4 Настройка среды разработки VisualStudio 2022

Прежде чем приступить к работе, убедитесь, что все инструменты и компоненты, необходимые для дальнейшей настройки среды разработки, установлены. Проверить это можно на панели инструментов, нажав кнопку «Получить инструменты и компоненты» и просмотрев список установленных компонентов. Должны быть установлены следующие компоненты:

– .NET 6 SDK (или более поздней версии);

– ASP.NET Core Runtime;

– NuGet Package Manager.

Следующим шагом является создание проекта. После нажатия кнопки «Создать проект» выберите шаблон C# «Консольное приложение (.NETCore)». После создания проекта в правой части экрана появится компонент интерфейса «Обозреватель решений», как показано на рисунке 12.

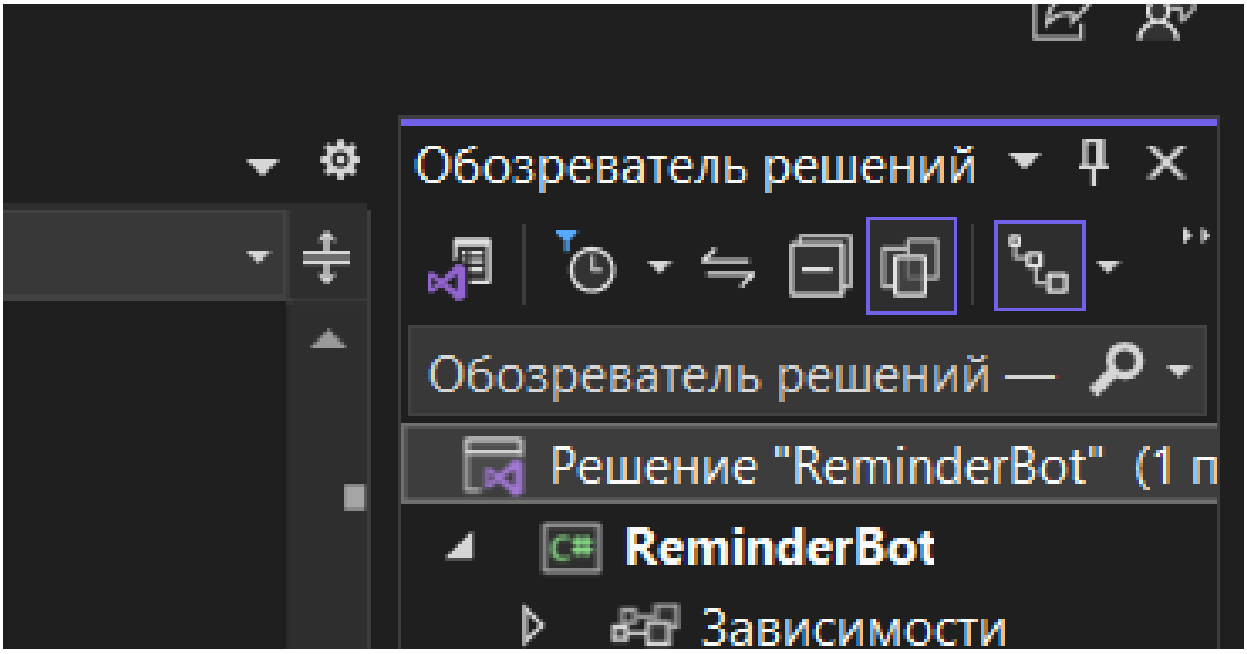


Рисунок 12 – Обозреватель решений

Щелкните по нему правой кнопкой мыши и выберите «Управление пакетами NuGet для решения…», как показано на рисунке 13.

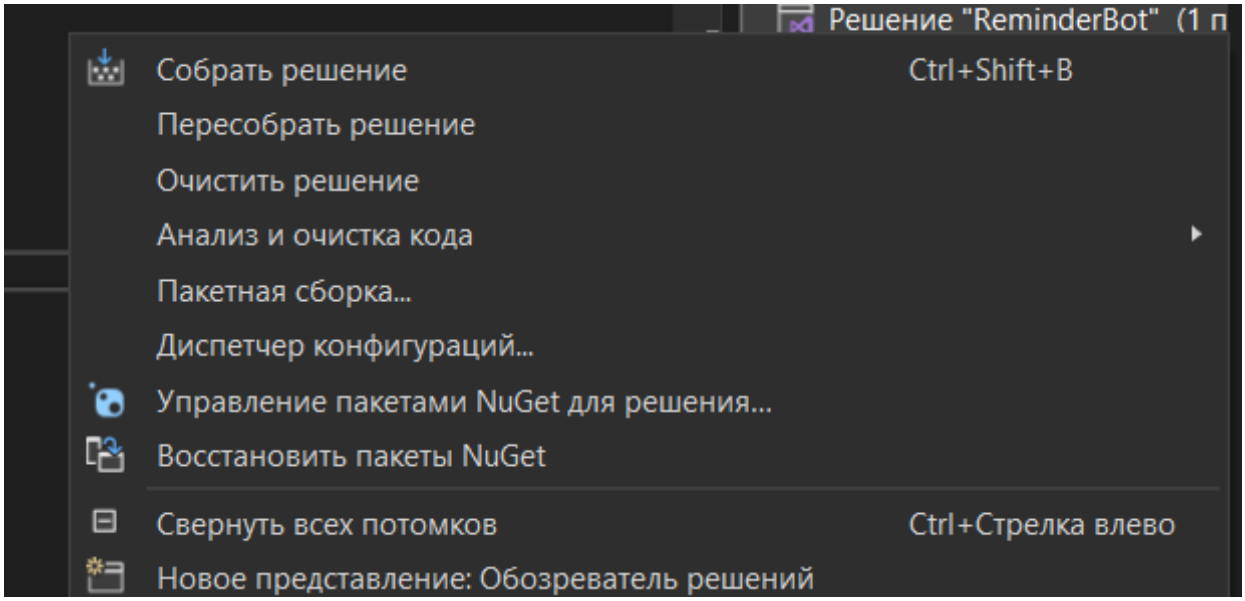


Рисунок 13 – Меню инструментов решений проекта

Ниже показан интерфейс «NuGet – Решения», который позволяет устанавливать или обновлять различные библиотеки, конфигурации API и другие компоненты, необходимые для бота «ВКонтакте», как показано на рисунке 14.

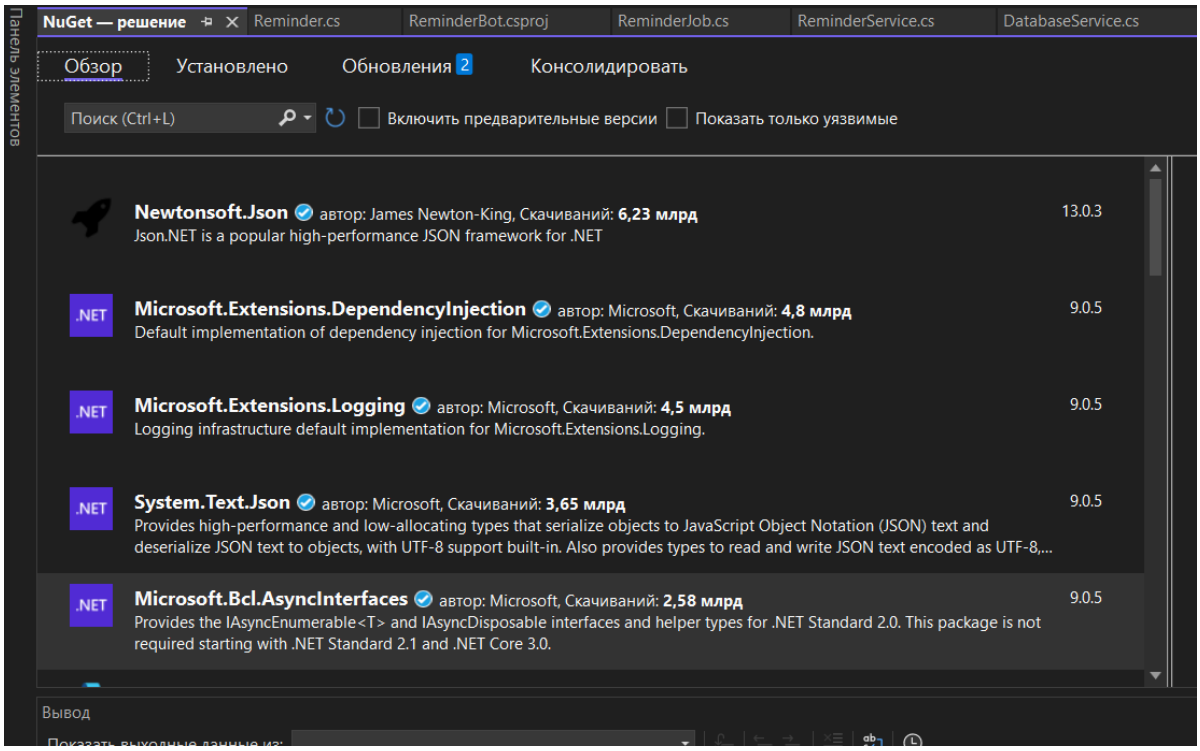


Рисунок 14 – Интерфейс «NuGet – решения»

В поисковой строке нужно ввести VkNet и установить эту библиотеку (рисунок 15).

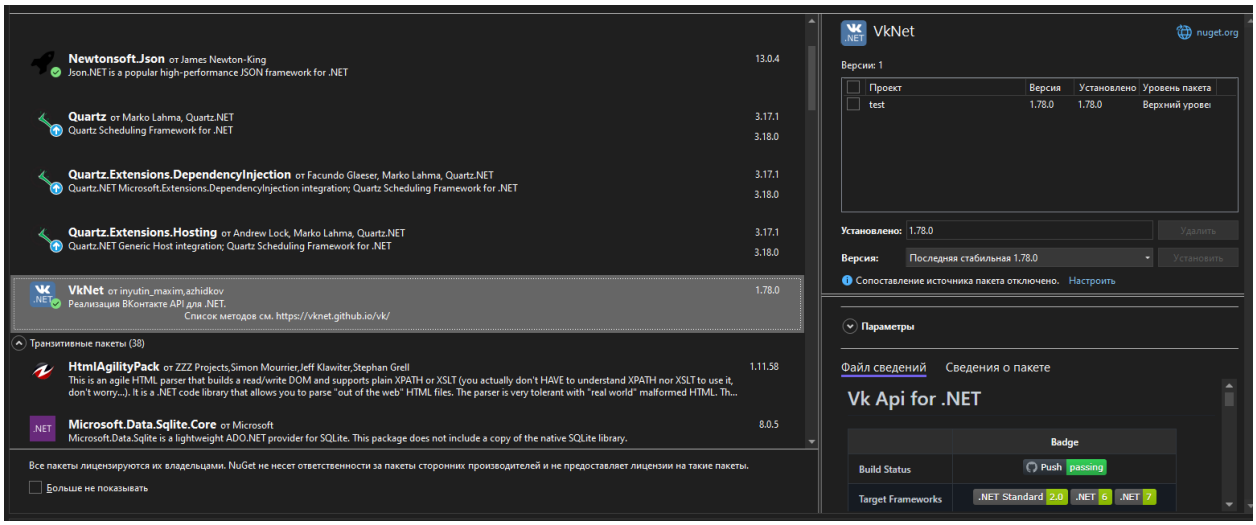


Рисунок 15 – Установка VkNet – библиотеки

VkNet представляет собой .NET-инструментарий, разработанный для взаимодействия с VKAPI версии 5.x. Эта библиотека обеспечивает:

– полную функциональность VKAPI.

– асинхронный подход к выполнению операций (с использованием async/await).

– поддержку обмена сообщениями, пользовательских клавиатур и управления сообществами.

– кроссплатформенность, совместимую с .NET 6, 7 и 8.

Существуют и другие варианты работы с VKAPI, включая:

– непосредственное взаимодействие с VKAPI через HTTP-запросы, что требует самостоятельной разработки логики обработки.

В контексте данного проекта выбор пал на VkNet из-за его простоты в использовании и полной пригодности для создания ботов.

Для организации локального хранения данных бота, таких как напоминания и пользовательские настройки, будет использована база данных Microsoft.Data.Sqlite. Эта технология привлекательна своей легковесностью и способностью связывать базы данных с объектно-ориентированным программированием, фактически создавая "виртуальную объектную базу данных" для работы с SQLite в .NET-приложениях.

Ключевыми преимуществами, определившими выбор именно этой библиотеки, являются:

– поддержка асинхронных операций (async/await) для всех взаимодействий с базой данных.

– кроссплатформенность, обеспечивающая работу на любых операционных системах.

– бесшовная интеграция с EntityFrameworkCore.

– поддержка транзакций и миграций базы данных.

Данная библиотека представлена на рисунке 16.

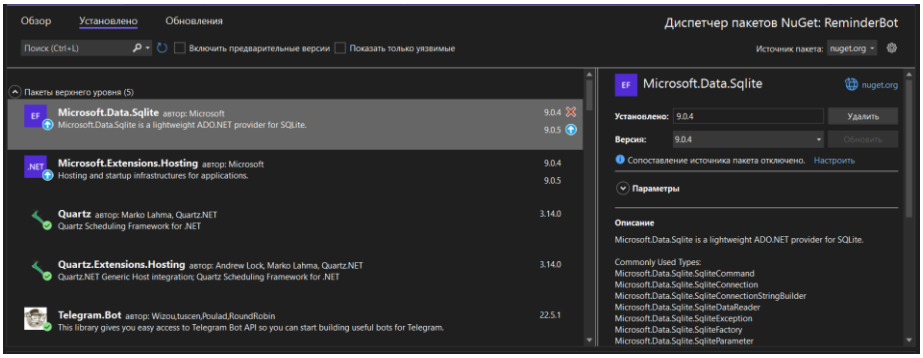


Рисунок 16 – База данных Microsoft.Data.Sqlite

Следующим шагом была произведена установка библиотеки Microsoft.Extensions.Hosting для создания управляемых хостов (hostedservices) в .NET.

Главные возможности включают в себя:

– главный хост приложения (IHost) – управляет жизненным циклом;

– фоновые сервисы (IHostedService) – для долгоживущих задач (например, обработка сообщений бота);

– встроенная DI (внедрение зависимостей) – не нужно подключать отдельные контейнеры;

– конфигурация — поддержка appsettings.json, переменных окружения.

А также используется для фоновых задач, в нашем случае для Quartz-планировщика и для консольных приложений с долгим сроком жизни (боты, демоны). На эту библиотеку можно посмотреть на рисунке 17.

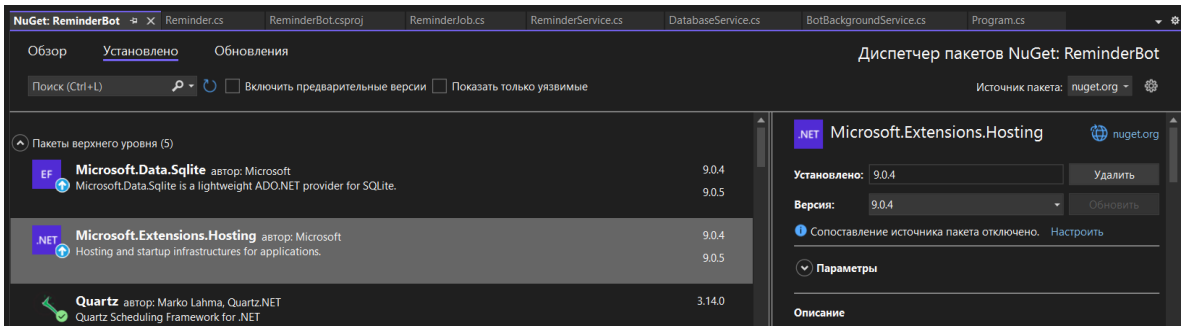


Рисунок 17 – Библиотека Microsoft.Extensions.Hosting

Для реализации планирования задач был выбран и интегрирован фреймворк Quartz, обладающий функциональностью работы с cron-выражениями и возможностью настройки весьма замысловатых триггеров, что наглядно демонстрируется на иллюстрации 18.

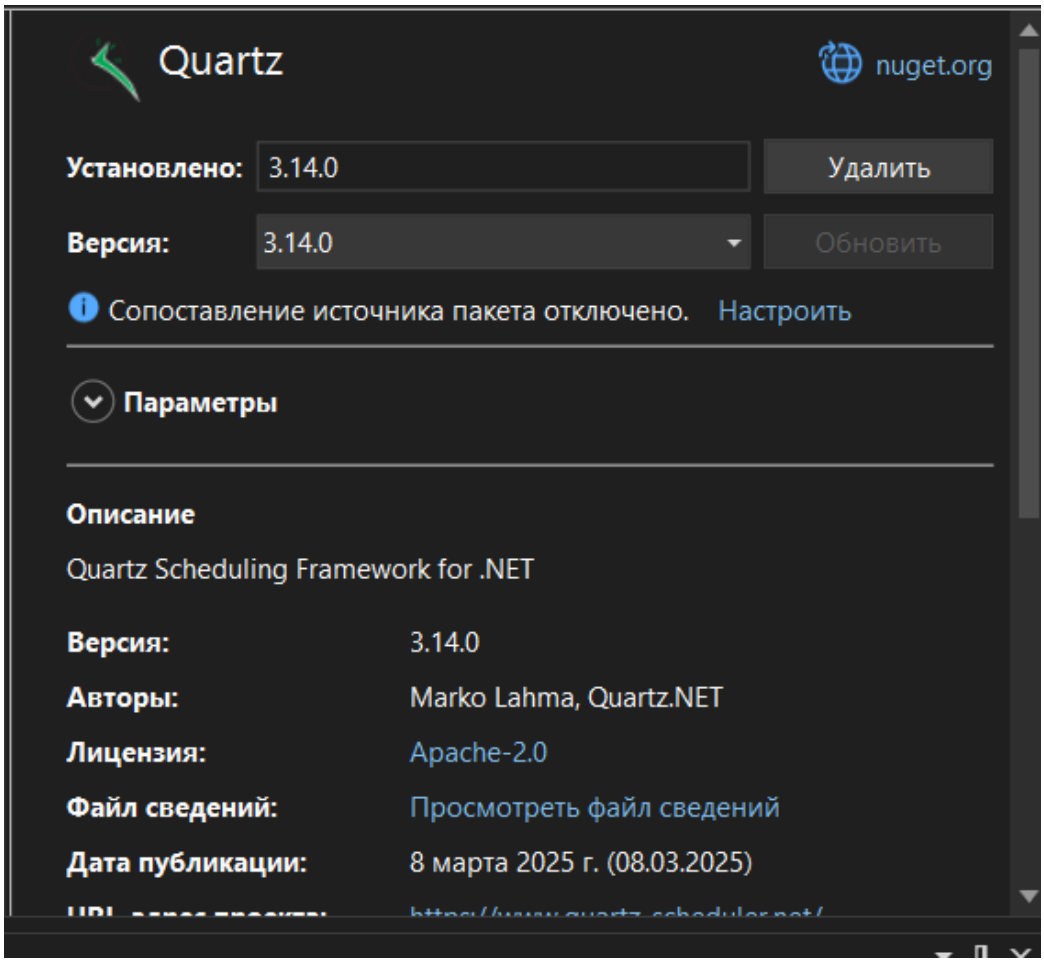


Рисунок 18 – Библиотека Quartz - планировщик

В рамках разработки системы напоминаний была использована библиотека Quartz, представляющая собой специализированный программный инструмент для организации и управления выполнением отложенных и периодических задач. Применение данного решения позволяет реализовать механизм генерации уведомлений в строго определённые моменты времени, а также обеспечить поддержку повторяющихся сценариев,

включая ежедневные и интервальные оповещения, что является критически важным для рассматриваемого класса приложений.

Функциональные возможности библиотеки характеризуются высокой степенью гибкости и масштабируемости. В частности, поддерживается формирование расписаний различной сложности посредством использования cron-выражений, интервальных триггеров и одноразовых заданий.

Дополнительно реализован механизм кластеризации, обеспечивающий распределённое выполнение задач в рамках нескольких экземпляров приложения, что повышает отказоустойчивость и устойчивость системы к нагрузкам. Существенным преимуществом является также поддержка асинхронной модели выполнения, основанной на использовании конструкций async/await, что способствует более эффективному управлению ресурсами и повышению общей производительности.

С целью интеграции планировщика задач в архитектуру приложения, построенного на основе Microsoft.Extensions.Hosting, было использовано расширение Quartz.Extensions.Hosting. Данное расширение обеспечивает автоматизированную инициализацию и запуск планировщика при старте хост-приложения, а также реализует механизмы управления его жизненным циклом. В частности, гарантируется корректное завершение выполняемых задач при остановке приложения, что позволяет избежать потери данных и не консистентных состояний.

Установка и подключение соответствующих компонентов осуществляется посредством стандартного инструментария управления зависимостями – менеджера пакетов NuGet. Процесс установки подробно иллюстрирован на рисунке 19.

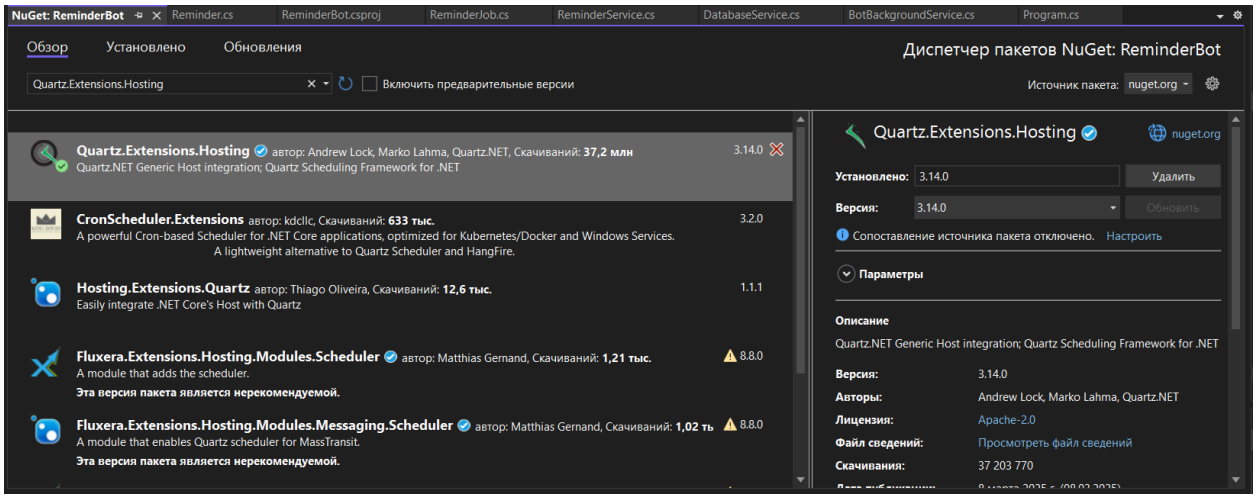


Рисунок 19 – Установка расширения Quartz.Extensions.Hosting

1.5.5 Настройка и подключение базы данных.

Перед началом разработки программной части приложения было выполнено проектирование и настройка системы хранения данных. В качестве системы управления базами данных была выбрана SQLite, что обусловлено её простотой, высокой производительностью и отсутствием необходимости в установке и администрировании отдельного серверного программного обеспечения.

Выбор SQLite также связан с особенностями разрабатываемого приложения, так как бот не предполагает высокой нагрузки и может эффективно работать с локальной файловой базой данных. Это позволяет упростить развертывание системы и снизить требования к инфраструктуре.

Подключение к базе данных реализовано с использованием библиотеки Microsoft.Data.Sqlite, обеспечивающей взаимодействие с SQLite в среде платформы .NET. В рамках инициализации приложения создаётся подключение к файлу базы данных reminders.db, который автоматически формируется при первом запуске программы.

На этапе подготовки базы данных была реализована автоматическая проверка существования необходимых таблиц. В случае их отсутствия выполняется создание таблицы Notifications, предназначенной для хранения

информации о напоминаниях. Данная таблица включает следующие поля: уникальный идентификатор записи, идентификатор пользователя, текст напоминания, дату и время его отправки, а также идентификатор задачи планировщика.

Таким образом, на этапе предварительной настройки была обеспечена готовность системы хранения данных к дальнейшей разработке, что позволило упростить реализацию функционала и обеспечить корректную работу механизма напоминаний.

2 Проектирование и разработка

Инициализация системы происходит при запуске приложения. Сначала создаётся хост, который настраивает и запускает все необходимые сервисы. Основу составляют несколько ключевых компонентов: клиент для работы с VK API, сервис для взаимодействия с базой данных, сервис управления напоминаниями и планировщик задач Quartz. Также активируется фоновый сервис для обработки сообщений от пользователей.

Листинг 1 – Создание хоста

```
var builder = Host.CreateDefaultBuilder(args);
```

Эта строка создаёт стандартный хост для .NET приложения. Метод Host.CreateDefaultBuilder() настраивает хост со стандартными параметрами, включая конфигурацию из appsettings.json, логирование и систему внедрения зависимостей.

Листинг 2 – Метод «ConfigureServices»

```
builder.ConfigureServices((context, services) =>
{
//Регистрация сервисов
});
```

Метод ConfigureServices используется для регистрации всех сервисов, которые будут доступны через систему внедрения зависимостей (DI).

Листинг 3 – Регистрация клиента VK API

```
services.AddSingleton<IVkMessageSender>(sp =>
    new VkMessageSender(sp.GetRequiredService<VkApi>(),
groupId));
```

Регистрируется сервис отправки сообщений в VK как Singleton. VkMessageSender отвечает за отправку текстовых сообщений и сообщений с клавиатурами.

Листинг 4 – Регистрация сервиса для базы данных

```
services.AddSingleton< NotificationStorageService>();
```

Регистрирует сервис для работы с базой данных как singleton, чтобы обеспечить единое подключение к БД на всё время работы приложения.

Листинг 5 – Регистрация основного сервиса напоминаний

```
services.AddSingleton< NotificationManagerService>();
```

Основной сервис для работы с напоминаниями также регистрируется как singleton.

Листинг 6 – Конфигурация планировщика задач

```
services.AddQuartz(quartz =>
{
    quartz.UseSimpleTypeLoader();
    quartz.UseInMemoryStore();
});
```

Конфигурирует планировщик задач Quartz.NET:

– UseSimpleTypeLoader - простой загрузчик типов;

– UseInMemoryStore - хранение задач в памяти.

Листинг 7 – Регистрация Quartz как фонового сервиса

```
services.AddQuartzHostedService(q=>q.WaitForJobsToComplete = true);
```

Регистрирует Quartz как фоновый сервис с настройкой: «WaitForJobsToComplete = true», что означает ожидание завершения всех активных задач при остановке приложения.

Листинг 8 – Регистрация основного сервиса

```
services.AddHostedService< VkBotPollingService>();
```

Регистрирует основной фоновый сервис бота VkBotPollingService, который отвечает за:

– получение сообщений через Long Poll;

– обработку команд пользователей;

- координацию работы других сервисов.

2.1 Основные компоненты

2.1.1 VkBotPollingService

Первый компонент – VkBotPollingService. Это фоновый сервис, отвечающий за основную работу бота. Он запускает процесс получения обновлений через Long Poll API социальной сети ВКонтакте, обрабатывает входящие сообщения, управляет клавиатурами и перенаправляет запросы в соответствующие сервисы для дальнейшей обработки. При успешном запуске сервис выводит информацию о готовности бота к работе.

Листинг 9 – метод «ExecuteAsync»

```
protected override async Task ExecuteAsync(CancellationToken
stoppingToken)
{
/…/
}
```

Метод ExecuteAsync – основной рабочий цикл фонового сервиса бота. Это переопределенный метод из базового класса BackgroundService, который выполняется при запуске хоста. Он отвечает за:

– инициализацию Long Poll соединения с VK API;

– непрерывное получение обновлений от пользователей;

– обработку входящих сообщений;

– контроль жизненного цикла сервиса.

CancellationToken stoppingToken – токен отмены, который используется для корректного завершения работы сервиса при остановке приложения.

Листинг 10 – Получение сервера Long Poll

```
    varlongPollServer    =    await    _vkApi.Groups.GetLongPoll
ServerAsync(_groupId);
```

Выполняется запрос к VK API для получения данных сервера Long Poll (адрес, ключ и начальное значение ts).

Листинг 11 – Основной цикл получения обновлений

```
    var response = await httpClient.GetStringAsync(url, stoppingToken);
    var json = JObject.Parse(response);
```

Происходит отправка запроса к серверу Long Poll и обработка JSON-ответа, содержащего новые события (сообщения от пользователей).

Листинг 12 – Информация о запуске бота

```
Console.WriteLine($"▪ VK Бот запущен для группы {_groupId}. Ждём сообщения...");
```

После успешного запуска сервиса в консоль выводится сообщение, подтверждающее, что бот готов к приёму сообщений от пользователей.

2.1.2 NotificationManagerService

Второй важный компонент – NotificationManagerService. Это центральный сервис для работы с напоминаниями. Он выполняет добавление новых напоминаний с автоматическим определением даты и времени из текста пользователя, предоставляет функции управления списком напоминаний, включая их просмотр, удаление. Сервис интегрирован с планировщиком Quartz для установки времени срабатывания уведомлений. Он взаимодействует с базой данных для сохранения и получения информации, проверяет корректность вводимых данных и формирует ответы пользователю.

Листинг 13 – Метод «CreateNotificationAsync»

```
public async Task CreateNotificationAsync(long chatId, string userMessage)
{
    // ...
}
```

Метод CreateNotificationAsync отвечает за создание нового напоминания. Это асинхронный метод, который принимает:

– long chatId – идентификатор чата с пользователем;

– string messageText – текст сообщения с напоминанием и датой.

Листинг 14 – Парсинг даты и времени

```
if (!TryParseRussianDateTime(dateString, out var remindAt))
    {
        await _botClient.SendMessage(chatId, "❌Невернаядата.
Формат: 29 декабря 20:30");
        return;
    }
```

Данный блок выполняет парсинг даты и времени из текста сообщения:

– TryParseRussianDateTime преобразует строку с датой в русском формате в объект DateTime. В случае ошибки пользователю отправляется сообщение с примером правильного формата.

Листинг 15 – Создание задачи для планировщика

```
var job = JobBuilder.Create<NotificationJob>()
    .WithIdentity(jobId)
    .UsingJobData("chatId", chatId)
    .UsingJobData("text", text)
    .Build();
```

Создаётся задача для планировщика Quartz:

– JobBuilder.Create<NotificationJob>() – создает задачу типа NotificationJob;

– WithIdentity(jobId) – устанавливает уникальный идентификатор задачи;

– UsingJobData – передаёт необходимые параметры (идентификатор чата и текст напоминания);

– Build() – завершает конфигурацию задачи.

Листинг 16 – Сохранение напоминания в базу данных

```
await _storage.AddNotificationAsync(chatId, text, notifyTime, jobId);
```

Производится сохранение напоминания в базу данных через сервис NotificationStorageService. Передаются параметры: идентификатор пользователя, текст напоминания, время срабатывания и уникальный ключ задачи Quartz.

2.1.3 NotificationStorageService

Третий компонент – NotificationStorageService. Он отвечает за все операции с базой данных SQLite. В его функции входит создание и инициализация базы данных, формирование необходимых таблиц, выполнение всех основных операций с данными — добавление, чтение и удаление информации о напоминаниях. Сервис поддерживает асинхронные операции для эффективной работы, хранит полные данные о напоминаниях, включая идентификаторы пользователей, текст сообщений, время срабатывания и служебные ключи задач, а также обеспечивает надежность хранения информации и обработку возможных ошибок.

Листинг 17 – Метод «AddNotificationAsync»

```
public async Task AddNotificationAsync(long chatId, string text, DateTime notifyTime, string jobId)
{
    // ...
}
```

Метод AddNotificationAsync выполняет асинхронное сохранение напоминания в базу данных. Это ключевой метод для персистентного хранения данных о напоминаниях.

Параметры метода:

- long chatId: уникальный идентификатор пользователя;
- string text: текст напоминания;
- DateTime remindAt: дата и время срабатывания напоминания;

– string jobKey: уникальный идентификатор связанной задачи в планировщике Quartz.

Листинг 18 – СозданиеSQL-запроса

```
cmd.CommandText = @"
    INSERT INTO Notifications (ChatId, MessageText, NotifyAt,
JobIdentifier)
    VALUES (@chatId, @text, @notifyAt, @jobId)";
```

Создает параметризованный SQL-запрос для:

– вставки новой записи в таблицу Notifications;

– указания всех необходимых полей;

– безопасной передачи данных через параметры.

Листинг 19 – Настройка параметров запроса

```
cmd.Parameters.AddWithValue("@chatId", chatId);
cmd.Parameters.AddWithValue("@text", text);
cmd.Parameters.AddWithValue("@notifyAt",
notifyTime.ToString("O"));
cmd.Parameters.AddWithValue("@jobId", jobId);
```

Настраивает параметры запроса:

– «chatId»: привязывается идентификатор чата;

– «text»: привязывает текст напоминания;

– «notifyAt»: преобразует дату в строку формата ISO8601;

– «jobId»: привязывает ключ задачи Quartz.

Листинг 20 – Выполнение асинхронной SQL-команды

```
awaitcmd.ExecuteNonQueryAsync();
```

Асинхронно выполняется SQL-команда: «ExecuteNonQueryAsync» используется для операций INSERT, UPDATE и DELETE; ключевое слово await обеспечивает неблокирующее выполнение операции.

2.2 Тестирование

Для проведения тестирования VK-бота «Напоминания» были произведены следующие действия:

– функциональное тестирование;

– тестирование пользовательского интерфейса;

– тестирование обработки ошибок;

– интеграционное тестирование взаимодействия компонентов.

А также была поставлена цель тестирования, чтобы проверить корректность работы основных функций системы, включая:

– создание напоминаний с указанием текста и даты;

– хранение и отображение списка активных напоминаний;

– удаление напоминаний;

– отправку уведомлений в указанное время;

– работу пользовательского интерфейса.

В качестве сервера выступает персональный компьютер на операционной системе Windows. Запустив программу, в частности серверную часть на платформе Visual Studio 2022, открылось окно командной строки. В данном окне можно увидеть запущенные технологии для создания задач, а точнее библиотеку Quartz. Также для открытия хоста, был запущен Microsoft.Hosting. При успешном запуске сервера, мы наблюдаем уведомление, что бот запущен и готов к работе, что показано на рисунке 20.

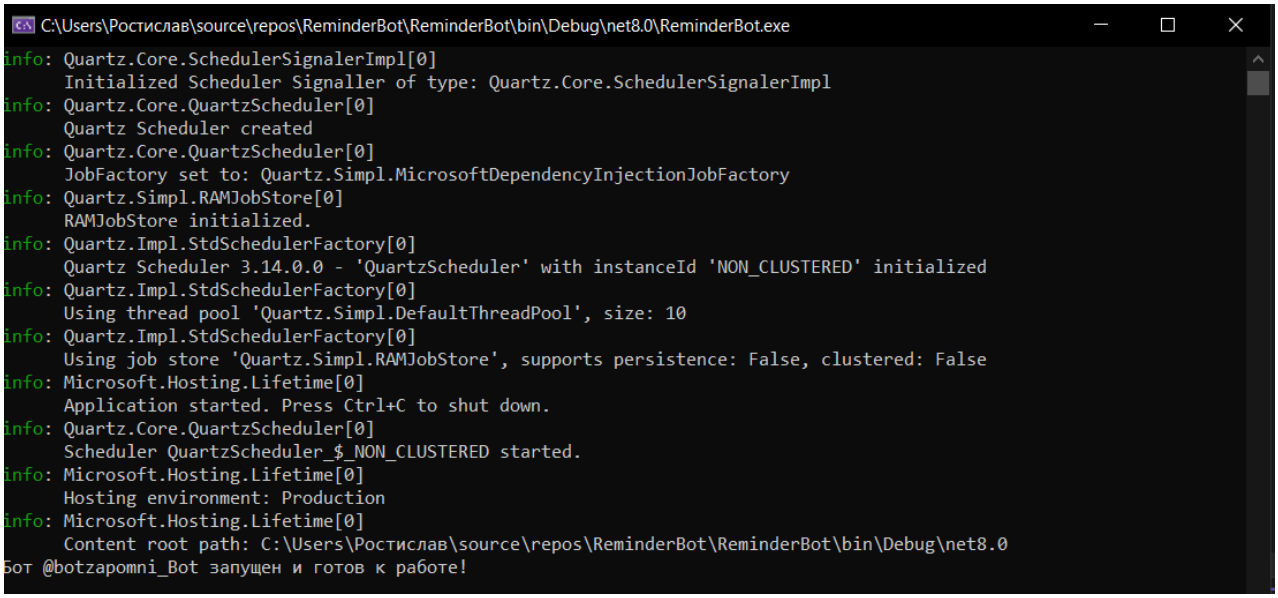


Рисунок 20 – Запуск серверной части чат-бота

После успешного запуска сервера открываем приложение ВКонтакте и переходим к непосредственному тестированию бота.

Тестирование проводилось на мобильном устройстве под управлением операционной системы iOS 18.6.2 с актуальной версией приложения VK 8.175.1.

Перед нами появляется минималистичный интерфейс бота с кнопкой «Начать» внизу экрана, которая позволяет приступить к работе с ботом (рисунок 21).

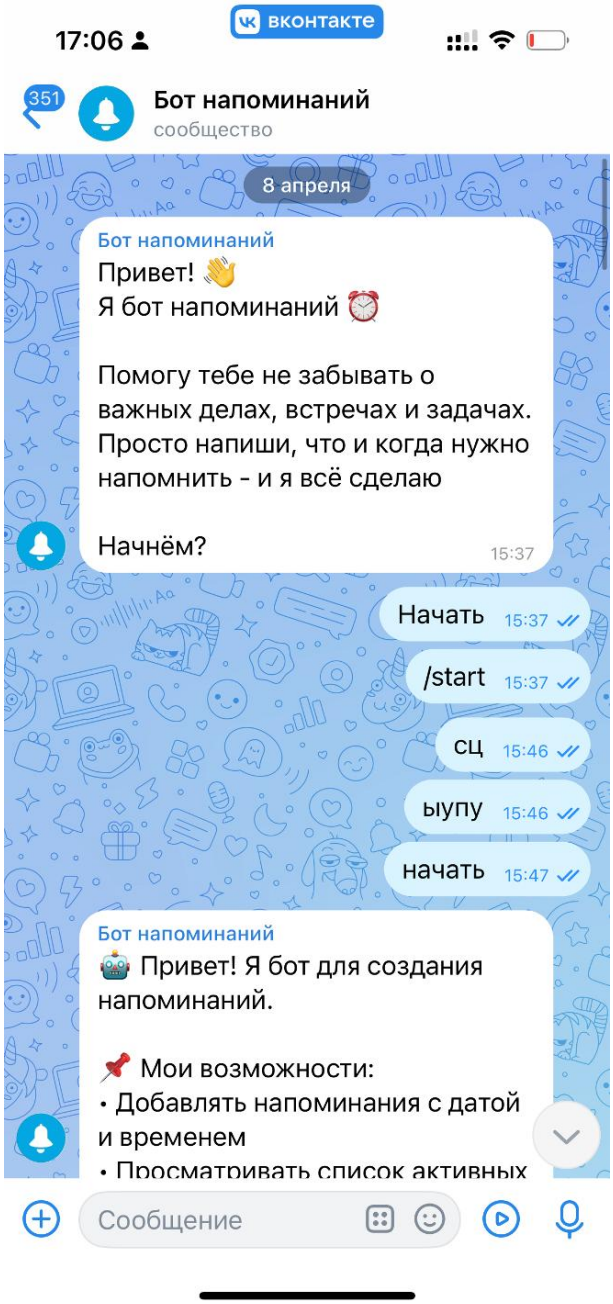


Рисунок 21 – Начальный интерфейс ВКонтакте-бота.

После нажатия кнопки «Начать» бот активируется и приветствует пользователя. На экране появляется информационное сообщение с описанием основных возможностей бота.

В нижней части интерфейса отображаются удобные кнопки быстрого доступа, что значительно упрощает взаимодействие с ботом. В современных

условиях важно, чтобы интерфейс был интуитивно понятным и не требовал ввода длинных команд вручную.

Для удобства пользователя в боте реализованы следующие кнопки:

– « ▪ Добавить напоминание»

– « ▪ Мои напоминания»

– «i Помощь»

Данный интерфейс представлен на рисунке 22.

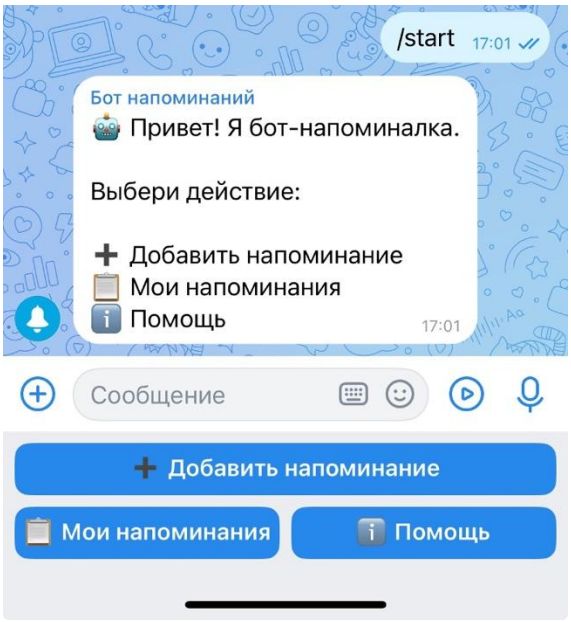


Рисунок 22 – Экран приветствия

После ознакомления с начальным интерфейсом переходим к созданию напоминания. Нажимаем кнопку « ▪ Добавить напоминание».

В ответ бот выводит сообщение с инлайн-кнопками, предлагая выбрать тип напоминания. Данная возможность позволяет пользователю быстро определить формат будущего уведомления. Эти интерфейсные элементы показаны на рисунке 23.

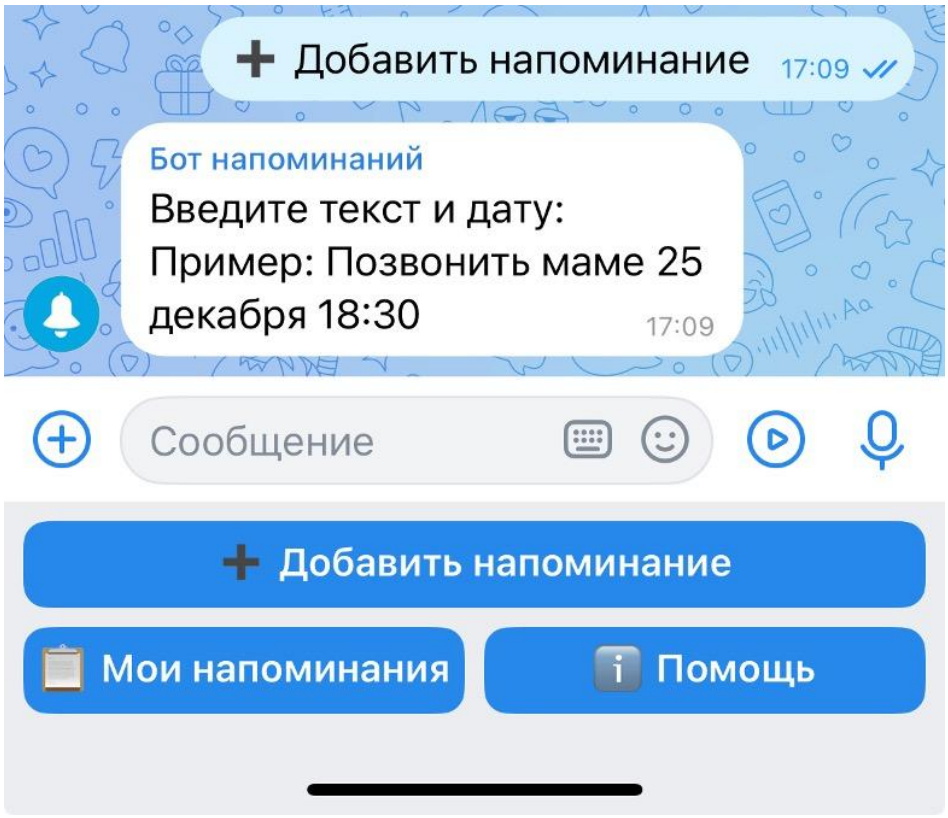


Рисунок 23 – Ознакомление с интерфейсом добавления напоминаний

Выбираем создание обычного напоминания. После нажатия соответствующей кнопки бот выводит подсказку с примером поддерживаемого формата ввода. Это помогает пользователю корректно составить сообщение без ошибок.

Также в нижней панели появляется кнопка «Назад», позволяющая в любой момент вернуться в главное меню.

После ознакомления с примером вводим текст напоминания вместе с датой и временем в указанном формате. Отправляем сообщение и получаем подтверждение от бота: «Напоминание успешно создано». В ответе также отображается текст напоминания и точное время, когда оно сработает. Данный процесс продемонстрирован на рисунке 24.

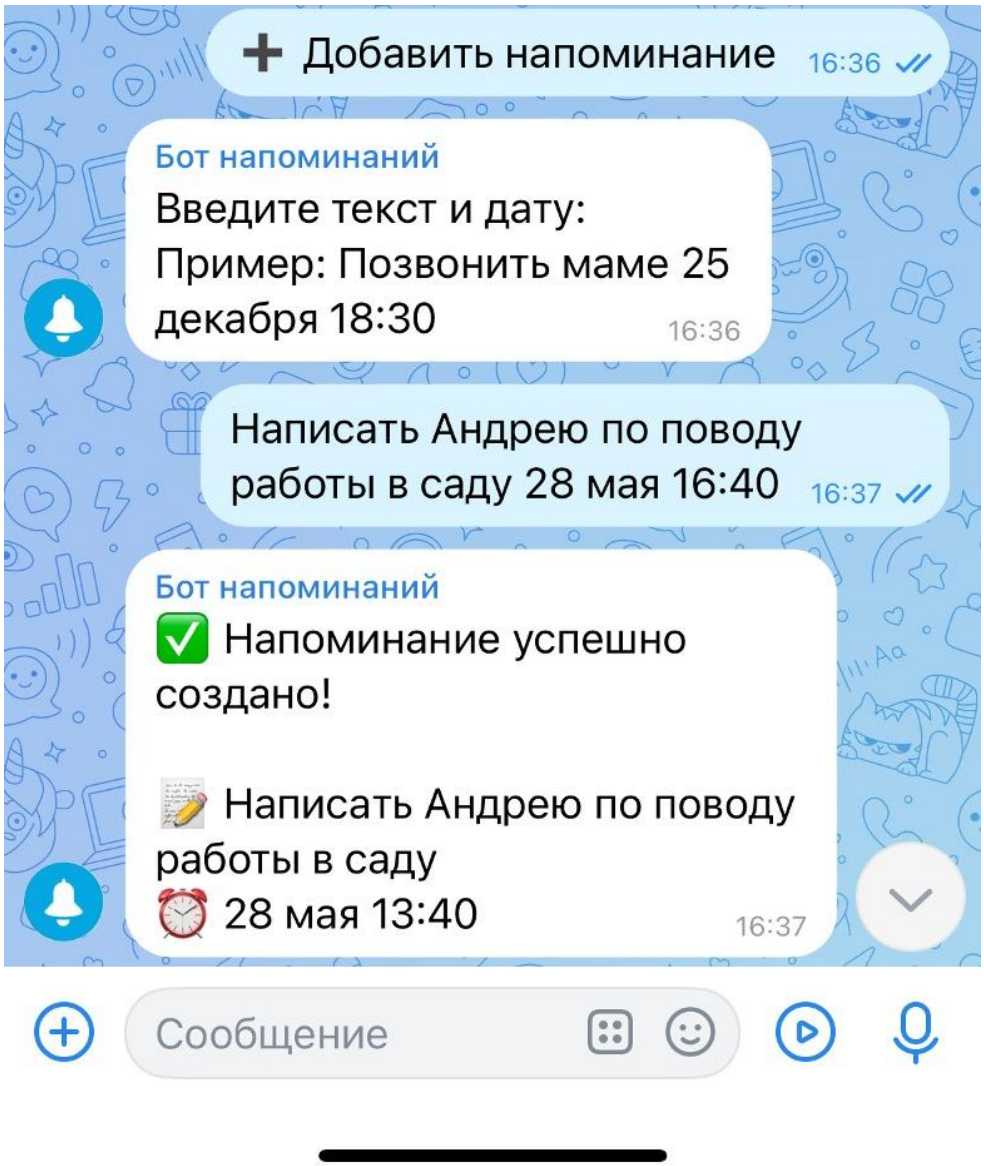


Рисунок 24 – Создание напоминание в ВКонтакте-боте

После создания напоминания можно просмотреть весь список активных записей, нажав кнопку « ▪ Мои напоминания».

Бот отправляет сообщение, в котором отображается полный перечень всех сохранённых напоминаний, которые ещё не были выполнены. Для каждого напоминания доступна кнопка удаления. Если пользователь передумал, он может нажать кнопку «Назад» и вернуться в главное меню.

Данный процесс продемонстрирован на рисунке 25.

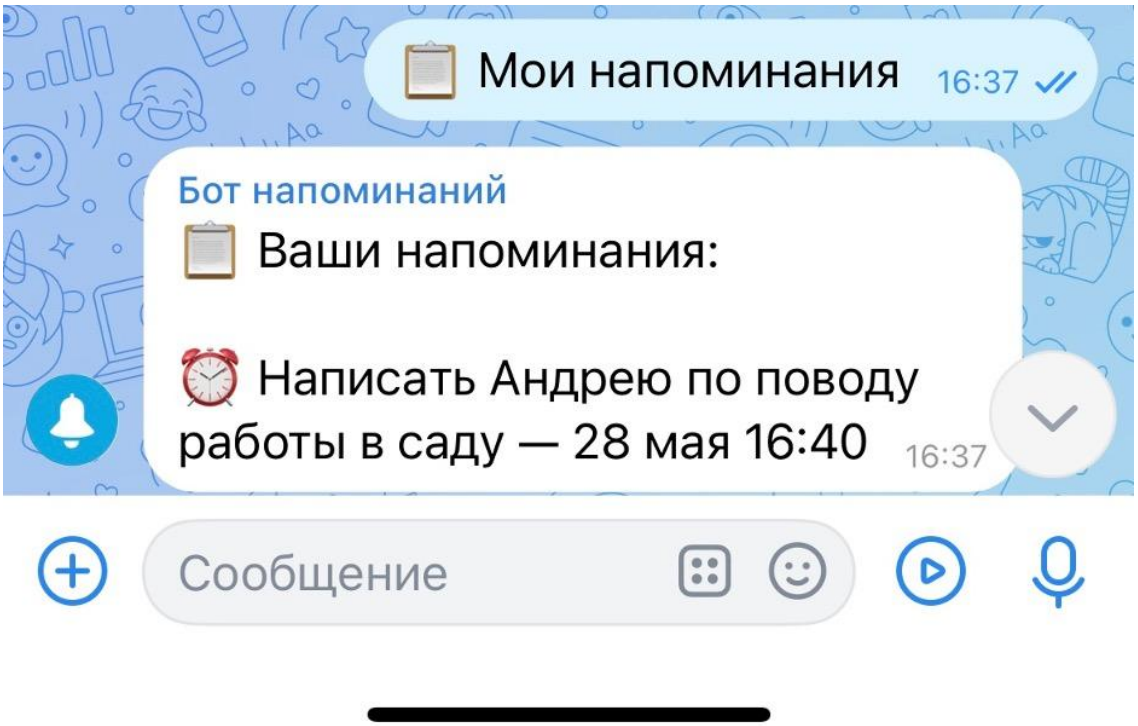


Рисунок 25 – Список напоминаний

Когда наступает указанная дата и время, бот автоматически отправляет уведомление с текстом, который был задан при создании напоминания. Уведомление появляется как в диалоге с ботом, так и на экране блокировки мобильного устройства.

После отправки напоминание автоматически удаляется из списка активных записей и из базы данных. Данный процесс продемонстрирован на рисунке 26.

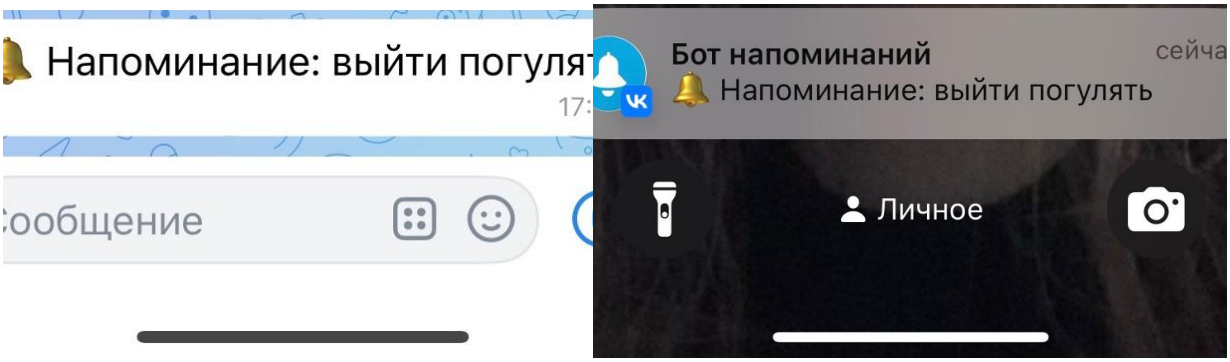


Рисунок 26 – Уведомление о напоминании в диалоге и экране уведомлений

Также было проведено тестирование расширенных типов напоминаний.

В частности, проверялась работа напоминаний с предварительным уведомлением (за несколько дней до события). Такой тип особенно удобен для важных событий, например, чтобы вовремя купить подарок на годовщину или не пропустить значимую дату.

Помимо этого, были протестированы повторяющиеся напоминания с различной периодичностью: каждый час, каждый день и каждую неделю. Данный функционал продемонстрирован на рисунках 27–29.

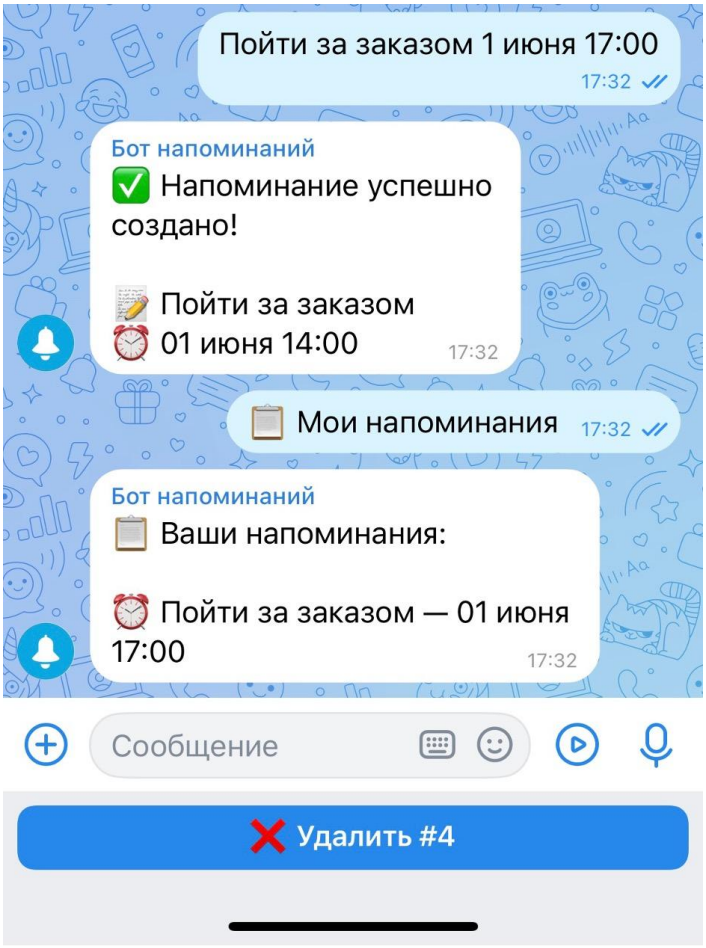


Рисунок 27 – Создание напоминаний с предупреждением за несколько дней

Вернувшись в главное меню бота, можно воспользоваться кнопкой «ℹ Помощь».

При нажатии открывается подробная инструкция по работе с ботом. В ней описаны основные аспекты использования:

– добавление новых напоминаний;

– просмотр списка активных напоминаний;

– удаление ненужных напоминаний;

– навигация по меню бота.

Данные возможности и внешний вид раздела помощи показаны на рисунке 30. Рекомендуется ознакомиться с этой инструкцией перед активным использованием бота.

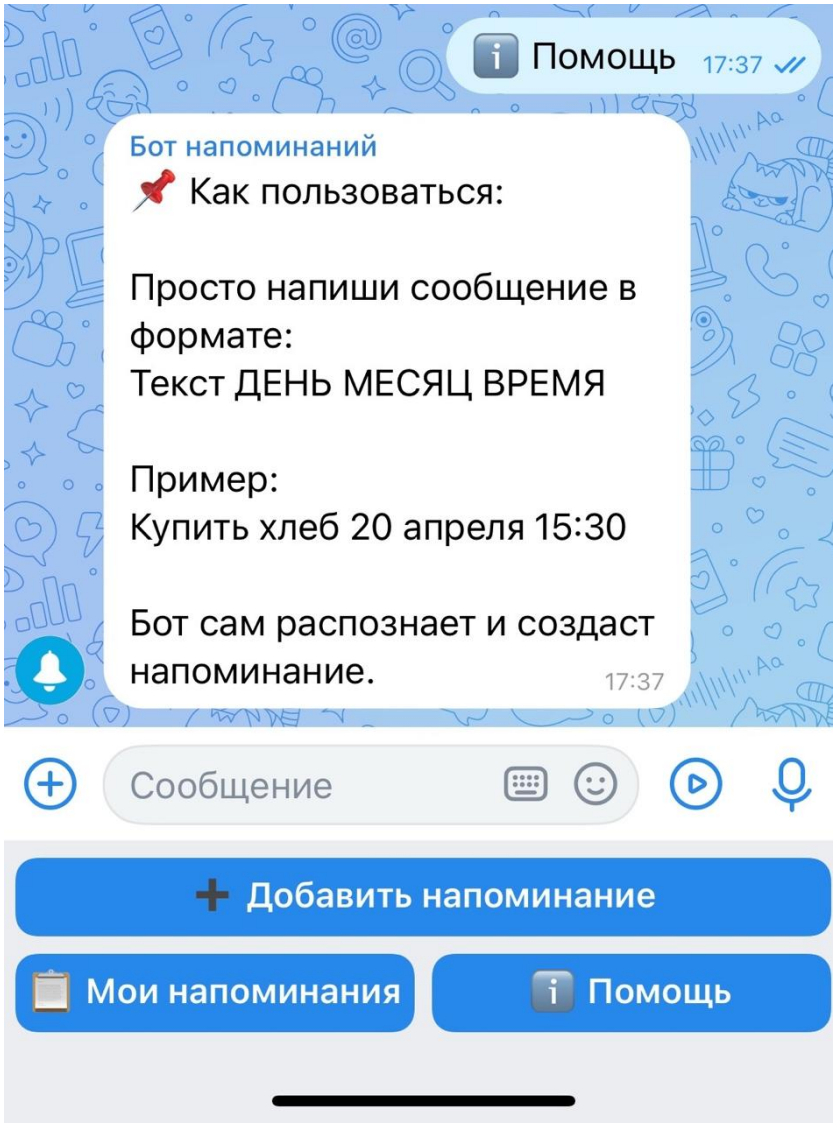


Рисунок 30 – Подробная инструкция по использованию бота

3 Безопасность жизнедеятельности

3.1 Правила безопасности при использовании ПК

Правила безопасной работы с персональным компьютером направлены на минимизацию основных профессиональных рисков, определённых в ГОСТ 12.0.003-2015. К наиболее значимым из них относятся:

– повышенная температура отдельных элементов системного блока и монитора;

– монотонность выполняемых операций;

– высокая зрительная нагрузка;

– возможность поражения статическим электричеством;

– недостаточная освещённость рабочего места;

– повышенный уровень электромагнитных полей в зоне работы пользователя;

– другие неблагоприятные факторы.

Полностью устранить влияние данных факторов невозможно, однако работодатель обязан принимать все необходимые меры по их снижению. Это позволяет сотрудникам восстанавливать нормальное самочувствие и работоспособность к началу следующей рабочей смены естественным путём, без использования дополнительных средств, таких как медикаментозная терапия или специальный массаж.

3.2 Правила безопасности при использовании смартфона

Для того, чтобы предотвратить возможные поломки и повреждения смартфона, зарядного устройства и сопутствующего оборудования, а также уменьшить риски возгорания или поражения электрическим током, необходимо внимательно соблюдать следующие условия:

запрещено самостоятельно вскрывать смартфон, зарядное устройство или другие компоненты;

– не сдавливайте, не деформируйте и не подвергайте смартфон, зарядное устройство и аксессуары сильным механическим воздействиям, колющим или режущим повреждениям;

– не закрывайте вентиляционные отверстия и зоны подключения, держите устройства подальше от открытого огня, горящего табака, а также избегайте попадания на них воды, влаги или других жидкостей;

– не используйте телефон и зарядное устройство вблизи воды или других жидкостей, не прикасайтесь к ним мокрыми руками;

– если телефон или зарядное устройство намокли, не пытайтесь сушить их в духовке, микроволновой печи, феном или другими нагревательными приборами;

– не роняйте, не бросайте и не сгибайте мобильный телефон и зарядное устройство;

– не используйте телефон и зарядное устройство в случае их неисправности, а также после падения или механического повреждения.

Рекомендуется соблюдать расстояние не менее 1 см между корпусом смартфона и телом пользователя.

Чтобы предотвратить риски (а также в соответствии с законодательством большинства стран), запрещается использовать телефон во время управления транспортным средством, работы с производственным оборудованием и при выполнении других потенциально опасных действий.

Не рекомендуется использовать и хранить телефон при температуре окружающей среды ниже 0°C и выше 40°C. Оптимальный диапазон температуры для хранения устройства составляет от 0°C до 40°C.

Во время интенсивной работы корпус телефона и зарядное устройство могут нагреваться. При длительном контакте с кожей это может вызвать дискомфорт или даже лёгкий ожог.

Телефоны поддерживают передачу сигналов LTE, WCDMA, GSM, Wi-Fi и Bluetooth, и могут создавать электромагнитные поля, которые потенциально способны влиять на работу кардиостимуляторов и других медицинских имплантов. Владельцам таких устройств рекомендуется соблюдать следующие меры предосторожности:

– не носите мобильный телефон в кармане, расположенном непосредственно рядом с кардиостимулятором или другим имплантированным медицинским устройством;

– при наличии вопросов о совместимости устройства проконсультируйтесь с лечащим врачом;

– если вы чувствуете, что телефон мешает работе медицинского устройства, немедленно выключите его и обратитесь к специалисту.

Меры предосторожности при использовании зарядного устройства:

– при отключении зарядного устройства от розетки тяните за корпус адаптера, а не за кабель;

– при использовании телефона во время зарядки убедитесь, что адаптер питания легко доступен и подключён к ближайшей розетке;

– подключайте к телефону только те устройства и кабели, которые имеют сертификацию USB-IF или прошли соответствующее тестирование;

– замену аккумулятора следует производить только в авторизованном сервисном центре. Использование неоригинальной или не соответствующей стандартам батареи может привести к возгоранию, взрыву или утечке электролита;

– если из аккумулятора начала вытекать жидкость, не допускайте её попадания в глаза, на кожу или одежду. При попадании в глаза немедленно промойте их чистой водой и обратитесь к врачу;

– не подвергайте протекающий аккумулятор воздействию огня, чтобы избежать взрыва или возгорания;

– утилизацию старого телефона и аккумулятора необходимо производить в соответствии с местным законодательством. Запрещается выбрасывать их вместе с бытовыми отходами.

Не используйте и не храните мобильный телефон в пыльных, влажных, загрязнённых или прокуренных помещениях, а также в местах с сильным магнитным полем. Это может привести к повреждению устройства или короткому замыканию.

Держите телефон подальше от источников сильного тепла и прямых солнечных лучей. Не оставляйте устройство в автомобиле, особенно на приборной панели или в местах, где температура может превышать 60°C. Перегрев может привести к выходу из строя устройства, повреждению аккумулятора, взрыву или возгоранию.

Не используйте и не храните телефон вблизи легковоспламеняющихся или взрывоопасных веществ (на заправочных станциях, химических складах и т.п.). Также запрещается использование телефона во время взрывных работ или в других потенциально опасных зонах. Одна искра в таких условиях может привести к взрыву, пожару и тяжёлым последствиям.

Заключение

В ходе выполнения выпускной квалификационной работы был проведён анализ существующих решений для управления напоминаниями. Было установлено, что VK-боты обладают значительными преимуществами, среди которых высокая доступность, простота использования и тесная интеграция с социальной сетью ВКонтакте. В то же время они уступают специализированным приложениям по ряду функциональных возможностей, в частности по синхронизации с внешними календарями и расширенной аналитике.

В связи с этим был разработан и успешно протестирован собственный VK-бот «Напоминания». Созданный бот демонстрирует стабильную и эффективную работу по основным задачам: создание напоминаний различного типа, их надёжное хранение, а также своевременную отправку уведомлений пользователю.

Модульная архитектура системы, включающая такие ключевые компоненты, как VkBotPollingService, NotificationManagerService, NotificationStorageService и NotificationJob, обеспечивает высокую гибкость, удобство сопровождения и дальнейшего развития проекта.

Проведённое тестирование подтвердило корректность реализации всех заявленных функций: обработку пользовательских команд, взаимодействие с базой данных, работу планировщика задач Quartz.NET, а также удобство пользовательского интерфейса. Интерфейс бота был оценён как интуитивно понятный и соответствующий современным требованиям к подобным приложениям.

В дальнейшем планируется расширение функциональности бота, в частности добавление поддержки голосовых сообщений, интеграция с другими сервисами и повышение интеллектуальных возможностей системы.

Таким образом, поставленные в работе цели и задачи были полностью достигнуты. Разработанный бот может служить как готовым решением для

повседневного использования, так и основой для дальнейшего развития и совершенствования.

Список использованных источников

1. Чат-боты, и как нам с этим поможет Microsoft. URL: https://habr.com/ru/companies/pvs-studio/articles/311592/(дата обращения 06.05.2026)

2. Todoist. URL: https://www.todoist.com/ru(дата обращения 09.05.2026)

3. Google Keep. URL: https://keep.google.com/(дата обращения 09.05.2026)

4. MicrosoftToDo. URL: https://to-do.office.com/tasks/ (дата обращения 09.05.2026)

5. VK BOT C#.URL:https://ru.stackoverflow.com/questions/1562257/telegram-bot-c (дата обращения 11.05.2026)

6. Quartz 3 Quick Start. URL: https://www.quartzscheduler.net/documentation/quartz-3.x/quick-start.html (дата обращения 16.05.2026)

7. Обзор Microsoft.Data.Sqlite. URL: https://learn.microsoft.com/ru-ru/dotnet/standard/data/sqlite/?tabs=net-cli (дата обращения 16.05.2026)

8. Visual Studio 2022. URL: https://visualstudio.microsoft.com/ru/vs/ (дата обращения 16.05.2026)

12. MicrosoftLearn – Официальная документация по C#, .NET и BotFramework. URL: https://learn.microsoft.com/ru-ru/azure/bot-service/bot-service-quickstart-create-bot?view=azure-bot-service-4.0&tabs=csharp%2Cvs(дата обращения 18.05.2026)

13.MicrosoftDocs – Официальная документация. URL: https://learn. microsoft.com/en-us/dotnet/fundamentals/ (дата обращения 25.05.2026)

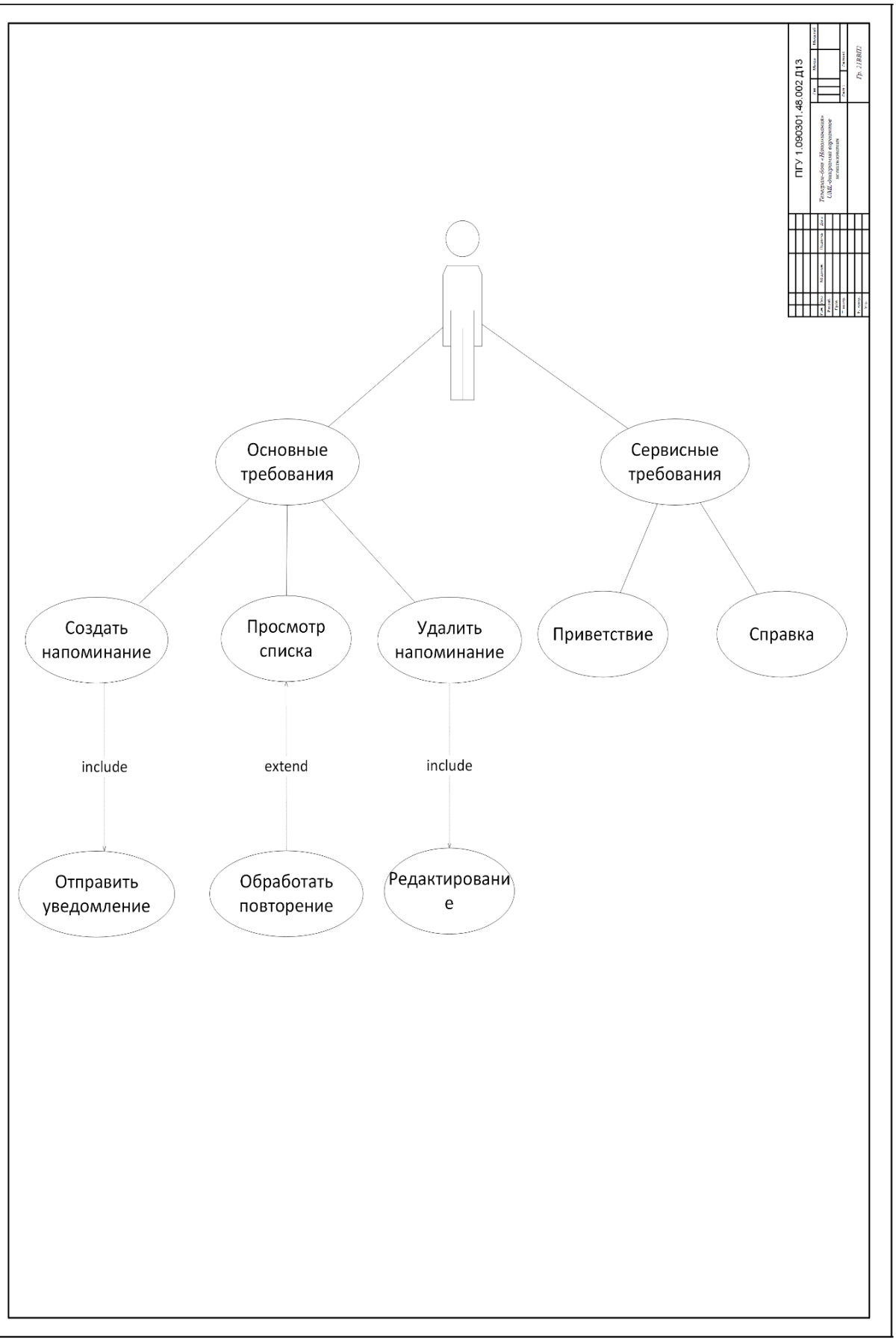
Приложение А

(обязательное)

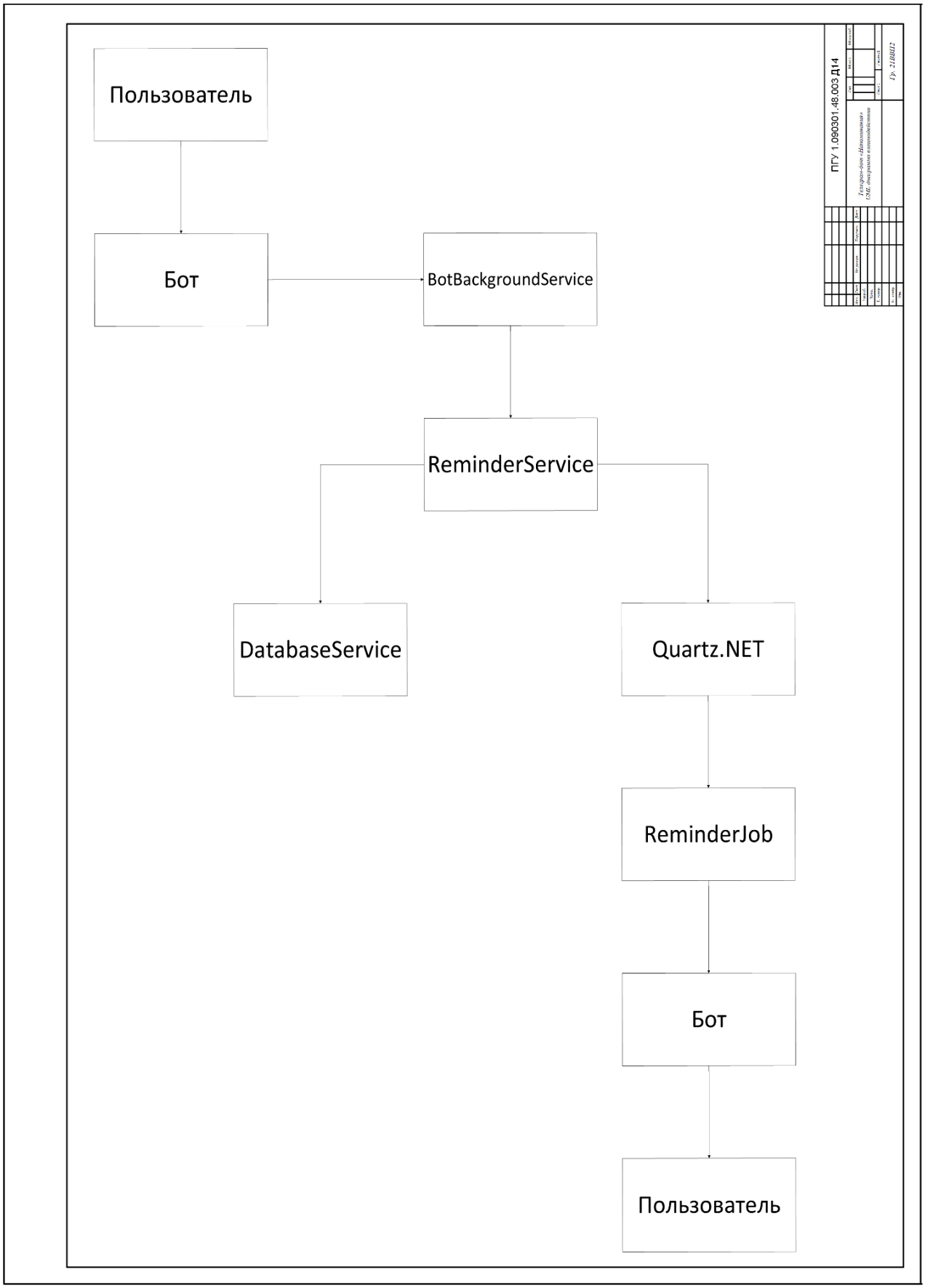
ВКонтакте чат-бот «Напоминания»

UML–диаграммы

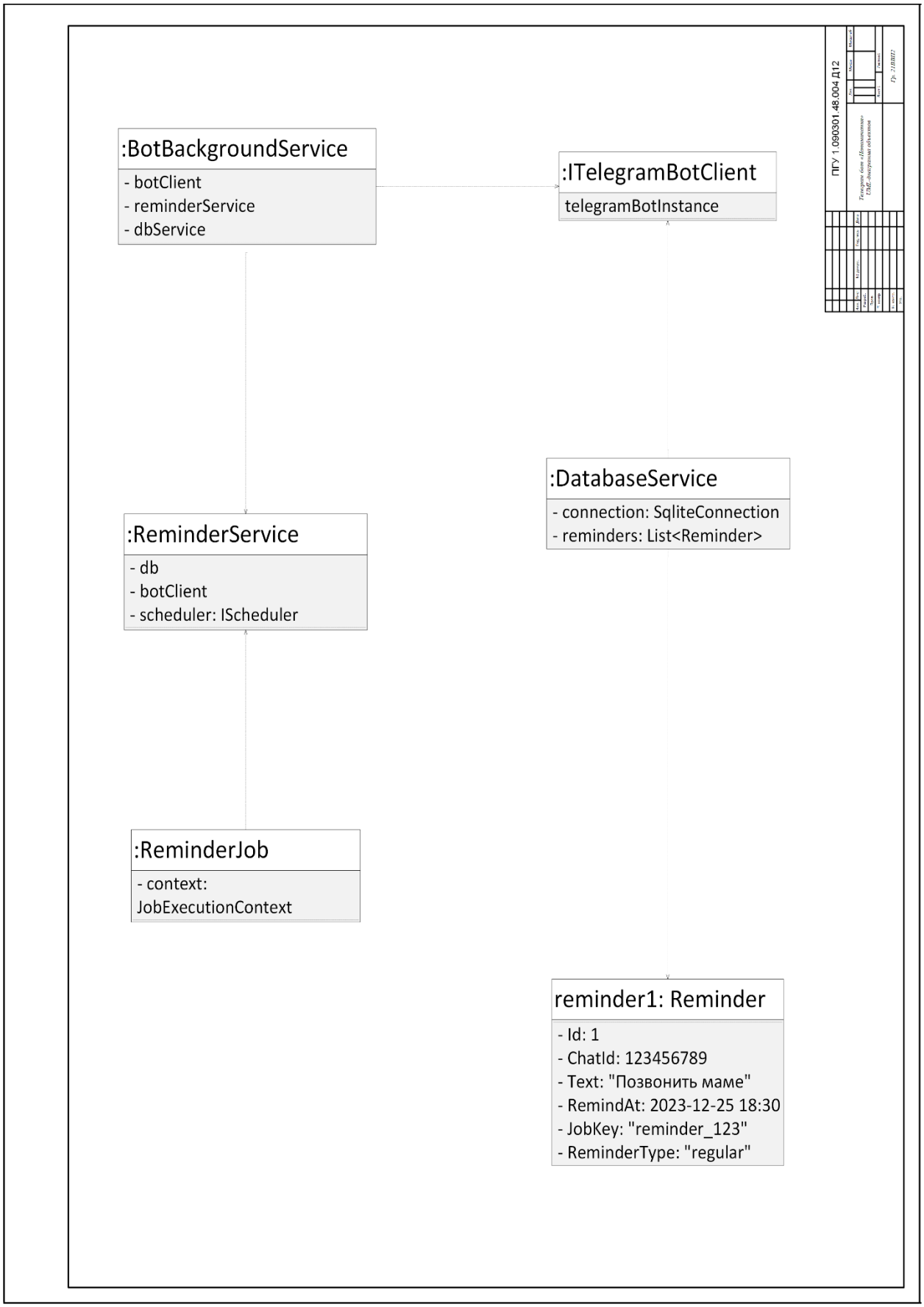
UML– диаграмма вариантов использования



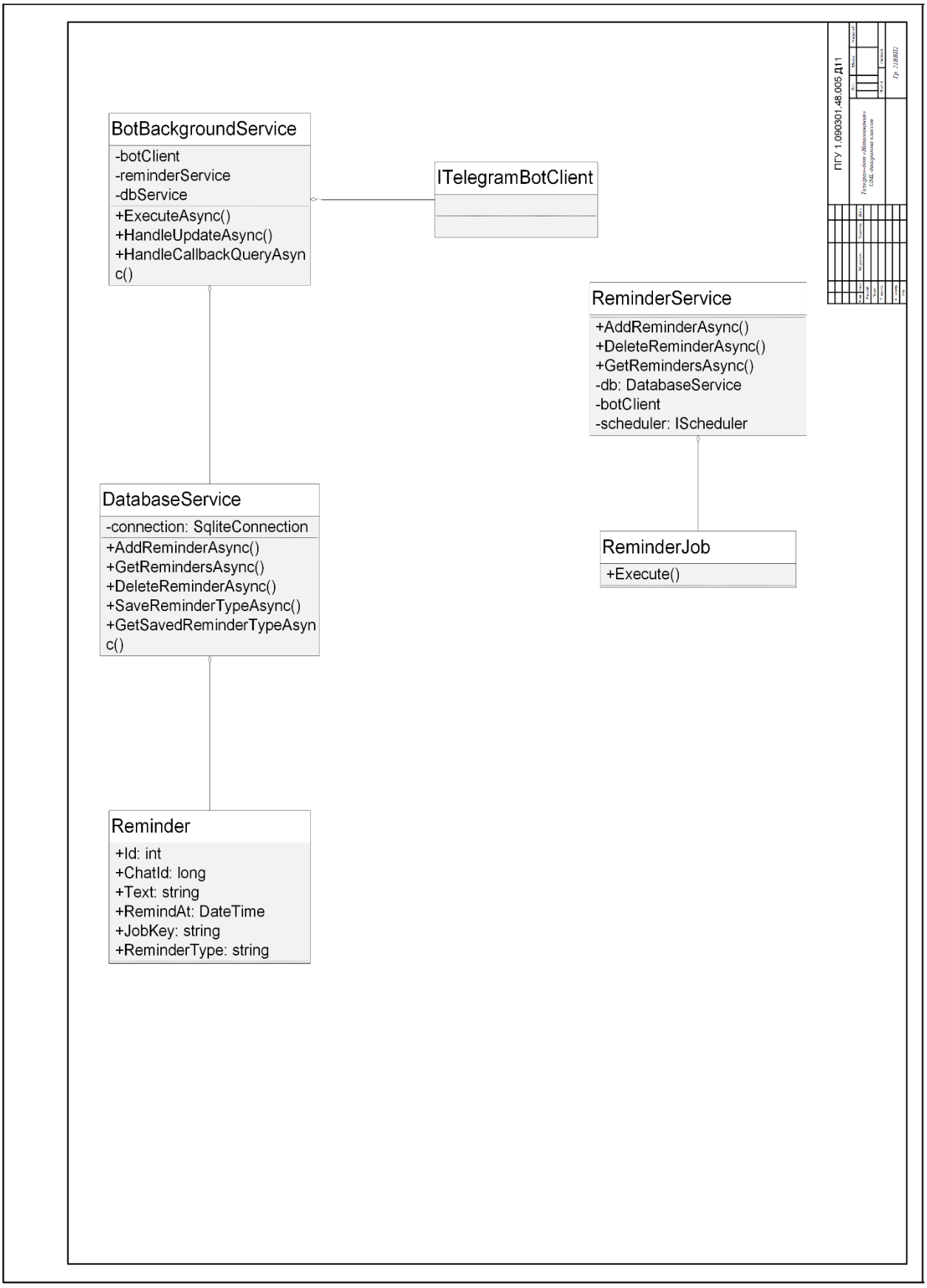
UML–диаграмма взаимодействия компонентов



UML–диаграмма объектов



UML– диаграмма классов



Приложение Б

(обязательное)

«ВКонтакте чат-бот “Напоминания”»

Листинг программы

Файл «Program.cs»

```
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Configuration;
using Quartz;
using VkNet;
using VkNet.Model;

namespace ReminderVkBot;

internal class Program
{
    private static async Task Main(string[] args)
    {
        var host = CreateHostBuilder(args).Build();

        await host.StartAsync();

        // Инициализация планировщика после запуска хоста
        var notificationManager =
host.Services.GetRequiredService<NotificationManagerService>();
        await notificationManager.PrepareSchedulerAsync();

        await host.WaitForShutdownAsync();
    }

    private static IHostBuilder CreateHostBuilder(string[] args)
    {
```

```
        return Host.CreateDefaultBuilder(args)
            .ConfigureAppConfiguration(SetupConfiguration)
            .ConfigureServices(RegisterServices);
    }

    private static void SetupConfiguration(HostBuilderContext
context, IConfigurationBuilder configBuilder)
    {
        configBuilder.AddJsonFile("appsettings.json", optional:
false, reloadOnChange: true);
    }

    private static void RegisterServices(HostBuilderContext
context, IServiceCollection services)
    {
        var config = context.Configuration;

        string? token = config["BotConfiguration:AccessToken"];
        string? groupIdStr = config["BotConfiguration:GroupId"];

        if (string.IsNullOrWhiteSpace(token))
            throw new InvalidOperationException("Токен доступа
VK не найден в конфигурации!");

        if (!ulong.TryParse(groupIdStr, out ulong groupId))
            throw new InvalidOperationException("Некорректный
GroupId в appsettings.json");

        var vkApiInstance = new VkApi();
```

```
        vkApiInstance.Authorize(new ApiAuthParams { AccessToken
= token });


        services.AddSingleton(vkApiInstance);


        services.AddSingleton<IVkMessageSender,
VkMessageSender>(sp =>
            new VkMessageSender(sp.GetRequiredService<VkApi>(),
groupId));


        services.AddSingleton<NotificationStorageService>();
        services.AddSingleton<NotificationManagerService>();


        services.AddScoped<NotificationJob>();


        services.AddQuartz(quartz =>
        {
            quartz.UseMicrosoftDependencyInjectionJobFactory();
            quartz.UseSimpleTypeLoader();
            quartz.UseInMemoryStore();
        });


        services.AddQuartzHostedService(options =>
options.WaitForJobsToComplete = true);


        services.AddHostedService(sp =>
            new VkBotPollingService(
                sp.GetRequiredService<IVkMessageSender>(),

sp.GetRequiredService<NotificationManagerService>(),
```

```
                sp.GetRequiredService<VkApi>(),
                groupId));
        }
    }
```

Файл «NotificationStorageService.cs»

```
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Configuration;
using Quartz;
using VkNet;
using VkNet.Model;

namespace ReminderVkBot;

internal class Program
{
    private static async Task Main(string[] args)
    {
        var host = CreateHostBuilder(args).Build();

        await host.StartAsync();

        // Инициализация планировщика после запуска хоста
        var notificationManager =
host.Services.GetRequiredService<NotificationManagerService>();
        await notificationManager.PrepareSchedulerAsync();

        await host.WaitForShutdownAsync();
```

```
    }

    private static IHostBuilder CreateHostBuilder(string[] args)
    {
        return Host.CreateDefaultBuilder(args)
            .ConfigureAppConfiguration(SetupConfiguration)
            .ConfigureServices(RegisterServices);
    }

    private static void SetupConfiguration(HostBuilderContext
context, IConfigurationBuilder configBuilder)
    {
        configBuilder.AddJsonFile("appsettings.json", optional:
false, reloadOnChange: true);
    }

    private static void RegisterServices(HostBuilderContext
context, IServiceCollection services)
    {
        var config = context.Configuration;

        string? token = config["BotConfiguration:AccessToken"];
        string? groupIdStr = config["BotConfiguration:GroupId"];

        if (string.IsNullOrWhiteSpace(token))
            throw new InvalidOperationException("Токен доступа
VK не найден в конфигурации!");

        if (!ulong.TryParse(groupIdStr, out ulong groupId))
```

```
            throw new InvalidOperationException("Некорректный
GroupId в appsettings.json");



        var vkApiInstance = new VkApi();

        vkApiInstance.Authorize(new ApiAuthParams { AccessToken
= token });



        services.AddSingleton(vkApiInstance);



        services.AddSingleton<IVkMessageSender,
VkMessageSender>(sp =>

            new VkMessageSender(sp.GetRequiredService<VkApi>(),
groupId));



        services.AddSingleton<NotificationStorageService>();

        services.AddSingleton<NotificationManagerService>();



        services.AddScoped<NotificationJob>();



        services.AddQuartz(quartz =>

        {

            quartz.UseMicrosoftDependencyInjectionJobFactory();

            quartz.UseSimpleTypeLoader();

            quartz.UseInMemoryStore();

        });



        services.AddQuartzHostedService(options =>
options.WaitForJobsToComplete = true);



        services.AddHostedService(sp =>
```

```
            new VkBotPollingService(
                sp.GetRequiredService<IVkMessageSender>(),
sp.GetRequiredService<NotificationManagerService>(),
                sp.GetRequiredService<VkApi>(),
                groupId));
    }
}
```

Файл « VkBotPollingService.cs»

```
using System.Text.RegularExpressions;
using Microsoft.Extensions.Hosting;
using Newtonsoft.Json.Linq;
using VkNet;
using VkNet.Model;

namespace ReminderVkBot;

public class VkBotPollingService : BackgroundService
{
    private readonly IVkMessageSender _messageSender;
    private readonly NotificationManagerService
_notificationManager;
    private readonly VkApi _vkApi;
    private readonly ulong _groupId;

    public VkBotPollingService(
        IVkMessageSender messageSender,
        NotificationManagerService notificationManager,
```

```
        VkApi vkApi,

        ulong groupId)

    {

        _messageSender = messageSender;

        _notificationManager = notificationManager;

        _vkApi = vkApi;

        _groupId = groupId;

    }



    protected override async Task ExecuteAsync(CancellationToken
stoppingToken)

    {

        Console.WriteLine($"VK Бот запущен для группы
{_groupId}. Ожидаем сообщения...");



        var longPollServer = await
_vkApi.Groups.GetLongPollServerAsync(_groupId);

        var longPollUrl =
$"{longPollServer.Server}?act=a_check&key={longPollServer.Key}&t
s={longPollServer.Ts}&wait=25";



        using var httpClient = new HttpClient();



        while (!stoppingToken.IsCancellationRequested)

        {

            try

            {

                var response = await
httpClient.GetStringAsync(longPollUrl, stoppingToken);

                var json = JObject.Parse(response);
```

```
                longPollUrl =
$"{longPollServer.Server}?act=a_check&key={longPollServer.Key}&t
s={json["ts"]}&wait=25";


                if (json["updates"] is JArray updates)

                {

                    foreach (var update in updates)

                    {

                        if (update["type"]?.ToString() ==
"message_new")

                        {

                            var msg =
update["object"]?["message"] ?? update["object"];

                            if (msg != null)

                            {

                                var peerId =
msg["peer_id"]?.Value<long>() ?? 0;

                                var text =
msg["text"]?.ToString() ?? "";

                                text = Regex.Replace(text,
@"\[club\d+\|.*?\]\s*", "").Trim();


                                bool hasOnlySticker =
msg["attachments"] is JArray att &&

                                                      att.Any(a
=> a["type"]?.ToString() == "sticker") &&

string.IsNullOrWhiteSpace(text);


                                if (hasOnlySticker)

                                {
```

```
                                    await
ShowMainMenuAsync(peerId);

                                }

                                else

                                {

                                    await
ProcessUserMessageAsync(peerId, text);

                                }

                            }

                        }

                    }

                }

                catch (Exception ex)

                {

                    Console.WriteLine($"LongPoll ошибка:
{ex.Message}");

                    await Task.Delay(5000, stoppingToken);



                    var newServer = await
_vkApi.Groups.GetLongPollServerAsync(_groupId);

                    longPollUrl =
$"{newServer.Server}?act=a_check&key={newServer.Key}&ts={newServ
er.Ts}&wait=25";

                }

            }

        }



    private async Task ProcessUserMessageAsync(long peerId,
string text)

    {
```

```
        try

        {

            if (string.IsNullOrWhiteSpace(text)) return;


            if (text == "/start" || text == "/menu" ||

                text.Contains("привет",
StringComparison.OrdinalIgnoreCase) ||

                text == "начать")

            {

                await ShowMainMenuAsync(peerId);

            }

            else if (text.Contains("Добавить напоминание") ||
text == "▪ Добавить напоминание")

            {

                await
_messageSender.SendTextMessageAsync(peerId,

                    "Введите текст и дату:\nПример: Позвонить
маме 25 декабря 18:30");

            }

            else if (text.Contains("Мои напоминания") || text ==
"▪ Мои напоминания")

            {

                await
_notificationManager.ShowUserNotificationsAsync(peerId);

            }

            else if (text.Contains("Помощь") || text == "i
Помощь")

            {

                await SendHelpAsync(peerId);

            }

            else if (text.StartsWith("▪ Удалить #"))
```

```
            {
                var idStr = text.Replace("▪ Удалить #",
"").Trim();
                if (int.TryParse(idStr, out int id))
                {
                    await
_notificationManager.RemoveNotificationByIdAsync(id, peerId);
                }
                else
                {
                    await
_messageSender.SendTextMessageAsync(peerId, "▪ Неверный номер
напоминания");
                }
            }
            else
            {
                // Пытаемся добавить напоминание, если текст
выглядит как команда
                await
_notificationManager.CreateNotificationAsync(peerId, text);
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Ошибка обработки сообщения:
{ex}");
            await _messageSender.SendTextMessageAsync(peerId, "▪
Произошла ошибка");
        }
    }
```

```
    private async Task ShowMainMenuAsync(long peerId)
    {
        var welcomeText = "▪ Привет! Я бот-напоминалка.\n\n" +
                          "Выбери действие:\n\n" +
                          "▪ Добавить напоминание\n" +
                          "▪ Мои напоминания\n" +
                          "i Помощь";

        var buttons = new List<IReadOnlyList<string>>
        {
            new List<string> { "▪ Добавить напоминание" },
            new List<string> { "▪ Мои напоминания", "i Помощь" }
        };

        await _messageSender.SendMessageWithButtonsAsync(peerId,
welcomeText, buttons);
    }

    private async Task SendHelpAsync(long peerId)
    {
        await _messageSender.SendTextMessageAsync(peerId,
            "▪ Как пользоваться:\n\n" +
            "Просто напиши сообщение в формате:\n" +
            "Текст ДЕНЬ МЕСЯЦ ВРЕМЯ\n\n" +
            "Пример:\n" +
            "Купить хлеб 20 апреля 15:30\n\n" +
```

```
            "Бот сам распознает и создаст напоминание.");
    }
}
```

Файл « VkMessageSender.cs»

```
using VkNet;
using VkNet.Model;
using VkNet.Enums.StringEnums;
using System;
using System.Collections.Generic;
using System.Threading.Tasks;

namespace ReminderVkBot;

public class VkMessageSender : IVkMessageSender
{
    private readonly VkApi _vkApiClient;
    private readonly ulong _communityId;

    public VkMessageSender(VkApi vkApiClient, ulong communityId)
    {
        _vkApiClient = vkApiClient;
        _communityId = communityId;
    }

    public async Task SendTextMessageAsync(long recipientId,
string message)
    {
```

```
        await _vkApiClient.Messages.SendAsync(new
MessagesSendParams
        {
            PeerId = recipientId,
            Message = message,
            RandomId = Random.Shared.NextInt64()
        });
    }

    public async Task SendMessageWithButtonsAsync(
        long recipientId,
        string message,
        IReadOnlyList<IReadOnlyList<string>> buttonRows)
    {
        var keyboard = BuildKeyboardMarkup(buttonRows);

        await _vkApiClient.Messages.SendAsync(new
MessagesSendParams
        {
            PeerId = recipientId,
            Message = message,
            RandomId = Random.Shared.NextInt64(),
            Keyboard = keyboard
        });
    }

    private MessageKeyboard?
BuildKeyboardMarkup(IReadOnlyList<IReadOnlyList<string>>? rows)
    {
```

```
        if (rows == null || rows.Count == 0)
            return null;

        var keyboardRows = new
List<List<MessageKeyboardButton>>();

        foreach (var row in rows)
        {
            var currentRow = new List<MessageKeyboardButton>();

            foreach (var label in row)
            {
                currentRow.Add(new MessageKeyboardButton
                {
                    Action = new MessageKeyboardButtonAction
                    {
                        Type = KeyboardButtonActionType.Text,
                        Label = label
                    },
                    Color = KeyboardButtonColor.Primary
                });
            }

            keyboardRows.Add(currentRow);
        }

        return new MessageKeyboard
        {
```

```
            Buttons = keyboardRows,
            Inline = false
        };
    }
}
```

Файл « IVkMessageSender.cs»

```
using System.Collections.Generic;
using System.Threading.Tasks;

namespace ReminderVkBot;

public interface IVkMessageSender
{
    Task SendTextMessageAsync(long recipientId, string message);

    Task SendMessageWithButtonsAsync(
        long recipientId,
        string message,
        IReadOnlyList<IReadOnlyList<string>> buttonRows);
}
```

Файл « NotificationItem.cs»

```
namespace ReminderVkBot;

public class NotificationItem
{
    public int Id { get; set; }
    public long ChatId { get; set; }
```

```
    public string Text { get; set; } = string.Empty;
    public DateTime NotifyAt { get; set; }
    public string JobId { get; set; } = string.Empty;
}
```

Файл « NotificationJob.cs»

```
using Quartz;

namespace ReminderVkBot;

public class NotificationJob : IJob
{
    private readonly IVkMessageSender _messageSender;
    private readonly NotificationStorageService _storage;

    public NotificationJob(IVkMessageSender messageSender,
NotificationStorageService storage)
    {
        _messageSender = messageSender;
        _storage = storage;
    }

    public async Task Execute(IJobExecutionContext context)
    {
        var jobId = context.JobDetail.Key.Name;
        var data = context.MergedJobDataMap;

        if (!data.ContainsKey("chatId") ||
!data.ContainsKey("text")) return;
```

```
        var chatId = data.GetLong("chatId");

        var text = data.GetString("text");


        try

        {

            await _messageSender.SendTextMessageAsync(chatId,
$"▪ Напоминание: {text}");

            await
_storage.RemoveNotificationByJobIdAsync(jobId);

            Console.WriteLine($"[Job] Напоминание {jobId}
отправлено");

        }

        catch (Exception ex)

        {

            Console.WriteLine($"[Job] Ошибка: {ex.Message}");

        }

    }

}
```

Файл « NotificationManagerService.cs»

```
using Quartz;

using System.Globalization;


namespace ReminderVkBot;


public class NotificationManagerService

{

    private readonly NotificationStorageService _storage;
```

```
    private readonly IVkMessageSender _messageSender;
    private readonly ISchedulerFactory _schedulerFactory;
    private IScheduler? _scheduler;

    public NotificationManagerService(
        ISchedulerFactory schedulerFactory,
        NotificationStorageService storage,
        IVkMessageSender messageSender)
    {
        _schedulerFactory = schedulerFactory;
        _storage = storage;
        _messageSender = messageSender;
    }

    public async Task PrepareSchedulerAsync()
    {
        _scheduler = await _schedulerFactory.GetScheduler();
        await _scheduler.Start();

        var allNotifications = await
_storage.GetAllNotificationsAsync();
        var now = DateTime.UtcNow;

        Console.WriteLine($"[Scheduler] Восстанавливаем
{allNotifications.Count} напоминаний после перезапуска");

        foreach (var notification in allNotifications.Where(n =>
n.NotifyAt > now))
        {
```

```
            var job = JobBuilder.Create<NotificationJob>()
                .WithIdentity(notification.JobId)
                .UsingJobData("chatId", notification.ChatId)
                .UsingJobData("text", notification.Text)
                .Build();

            var trigger = TriggerBuilder.Create()
                .WithIdentity($"{notification.JobId}_trigger")
                .StartAt(notification.NotifyAt)
                .Build();

            await _scheduler.ScheduleJob(job, trigger);
            Console.WriteLine($"[Scheduler] Восстановлено
напоминание {notification.JobId} на {notification.NotifyAt:yyyy-
MM-dd HH:mm}");
        }
    }

    public async Task CreateNotificationAsync(long chatId,
string userMessage)
    {
        try
        {
            if (userMessage.StartsWith("/addreminder"))
                userMessage =
userMessage["/addreminder".Length..].Trim();

            var parts = userMessage.Split(' ',
StringSplitOptions.RemoveEmptyEntries);
            if (parts.Length < 4)
```

```
            {
                await
_messageSender.SendTextMessageAsync(chatId,
                    "▪ Неверный формат.\nПример: Позвонить маме
25 декабря 18:30");
                return;
            }

            var text = string.Join(" ", parts.Take(parts.Length
- 3));
            var dateString = string.Join(" ",
parts.Skip(parts.Length - 3));

            if (!TryParseRussianDate(dateString, out var
notifyTime))
            {
                await
_messageSender.SendTextMessageAsync(chatId,
                    "▪ Не удалось распознать дату.\nПравильный
формат: 25 декабря 18:30");
                return;
            }

            var jobId = $"notify_{Guid.NewGuid()}";

            await _storage.AddNotificationAsync(chatId, text,
notifyTime, jobId);

            if (_scheduler == null)
                _scheduler = await
_schedulerFactory.GetScheduler();
```

```
            var job = JobBuilder.Create<NotificationJob>()
                .WithIdentity(jobId)
                .UsingJobData("chatId", chatId)
                .UsingJobData("text", text)
                .Build();

            var trigger = TriggerBuilder.Create()
                .WithIdentity($"{jobId}_trigger")
                .StartAt(notifyTime)
                .Build();

            await _scheduler.ScheduleJob(job, trigger);

            await _messageSender.SendTextMessageAsync(chatId,
                $"▪ Напоминание успешно создано!\n\n▪ {text}\n▪
{notifyTime:dd MMMM HH:mm}");

            Console.WriteLine($"[Scheduler] Создано новое
напоминание {jobId}");
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Ошибка создания напоминания:
{ex}");
            await _messageSender.SendTextMessageAsync(chatId, "▪
Произошла ошибка при создании напоминания");
        }
    }
```

```
    private bool TryParseRussianDate(string dateString, out
DateTime result)

    {

        result = DateTime.MinValue;

        try

        {

            var months = new Dictionary<string,
int>(StringComparer.OrdinalIgnoreCase)

            {

                {"января", 1}, {"февраля", 2}, {"марта", 3},
{"апреля", 4},

                {"мая", 5}, {"июня", 6}, {"июля", 7},
{"августа", 8},

                {"сентября", 9}, {"октября", 10}, {"ноября",
11}, {"декабря", 12}

            };



            var parts = dateString.Split(' ');

            if (parts.Length != 3) return false;



            if (!int.TryParse(parts[0], out var day)) return
false;

            if (!months.TryGetValue(parts[1], out var month))
return false;

            if (!TimeSpan.TryParse(parts[2],
CultureInfo.InvariantCulture, out var time)) return false;



            var mskZone =
TimeZoneInfo.FindSystemTimeZoneById("Russian Standard Time");

            var nowMsk =
TimeZoneInfo.ConvertTimeFromUtc(DateTime.UtcNow, mskZone);
```

```
            var localDateTime = new DateTime(nowMsk.Year, month,
day, time.Hours, time.Minutes, 0);


            if (localDateTime <= nowMsk)

                localDateTime = localDateTime.AddYears(1);


            result =
TimeZoneInfo.ConvertTimeToUtc(localDateTime, mskZone);

            return true;

        }

        catch

        {

            return false;

        }

    }


    public async Task ShowUserNotificationsAsync(long chatId)

    {

        try

        {

            var notifications = await
_storage.GetUserNotificationsAsync(chatId);


            if (notifications.Count == 0)

            {

                await
_messageSender.SendTextMessageAsync(chatId, "▪ У вас нет
активных напоминаний");

                return;

            }
```

```
            var text = "▪ Ваши напоминания:\n\n";

            var buttons = new List<IReadOnlyList<string>>();


            foreach (var n in notifications.OrderBy(x =>
x.NotifyAt))

            {

                text += $"▪ {n.Text} — {n.NotifyAt:dd MMMM
HH:mm}\n";

                buttons.Add(new List<string> { $"▪ Удалить
#{n.Id}" });

            }



            await
_messageSender.SendMessageWithButtonsAsync(chatId, text,
buttons);

        }

        catch (Exception ex)

        {

            Console.WriteLine($"Ошибка показа списка: {ex}");

            await _messageSender.SendTextMessageAsync(chatId, "▪
Не удалось загрузить список");

        }

    }



    public async Task RemoveNotificationByIdAsync(int id, long
chatId)

    {

        try

        {
```

```
            var all = await _storage.GetAllNotificationsAsync();

            var notification = all.FirstOrDefault(x => x.Id ==
id);



            if (notification == null)

            {

                await
_messageSender.SendTextMessageAsync(chatId, "▪ Такое напоминание
не найдено");

                return;

            }



            await _storage.RemoveNotificationByIdAsync(id);



            if (_scheduler != null)

                await _scheduler.DeleteJob(new
JobKey(notification.JobId));



            await _messageSender.SendTextMessageAsync(chatId, "▪
Напоминание удалено");

            Console.WriteLine($"[Scheduler] Удалено напоминание
#{id}");

        }

        catch (Exception ex)

        {

            Console.WriteLine($"Ошибка удаления: {ex}");

            await _messageSender.SendTextMessageAsync(chatId, "▪
Не удалось удалить напоминание");

        }

    }

}
```

Приложение В

(обязательное)

«ВКонтакте чат-бот “Напоминания”»

Презентация

VK-бот «Напоминания»

Выполнил: студент группы 22ВВП1
Дасаев Тимур Зягитович

Научный руководитель:
К.т.н., доцент кафедры ВТ
Слепцов Н.В.

Цель проекта

Разработка кроссплатформенного бота

Система напоминаний

Удобный пользовательский интерфейс

Надежное планирование задач

Основная цель проекта — предоставить пользователям надежный и удобный инструмент для управления своим расписанием, минимизируя риск забыть о важных встречах, задачах или событиях.

2

Актуальность

Популярность

VK является одним из ведущих мессенджеров с огромной пользовательской базой, что делает его идеальной платформой для разработки ботов.

Недостатки существующих решений

Многие стандартные приложения для напоминаний либо неудобны в использовании, либо перегружены избыточным функционалом, который отталкивает пользователей.

Кроссплатформенность и доступность

VK-бот предлагает простое, доступное и кроссплатформенное решение для управления напоминаниями, доступное на любом устройстве.

Актуальность проекта обусловлена возрастающей потребностью в эффективных инструментах для организации личного времени в современном мире. Мой бот способен решить эти проблемы, предлагая оптимальное сочетание функциональности и удобства.

3

Актуальность

Популярность

VK является одним из ведущих мессенджеров с огромной пользовательской базой, что делает его идеальной платформой для разработки ботов.

Недостатки существующих решений

Многие стандартные приложения для напоминаний либо неудобны в использовании, либо перегружены избыточным функционалом, который отталкивает пользователей.

Кроссплатформенность и доступность

VK-бот предлагает простое, доступное и кроссплатформенное решение для управления напоминаниями, доступное на любом устройстве.

Актуальность проекта обусловлена возрастающей потребностью в эффективных инструментах для организации личного времени в современном мире. Мой бот способен решить эти проблемы, предлагая оптимальное сочетание функциональности и удобства.

Целевая аудитория

Студенты

Для эффективного управления учебным расписанием, сроками сдачи работ и подготовки к экзаменам.

Работники офисов

Для организации рабочих встреч, дедлайнов по проектам и управления задачами.

Люди с плотным графиком

Для тех, кто нуждается в постоянном контроле над своим временем и не хочет упустить важные события.

Пожилые пользователи

Для упрощения повседневных задач, таких как прием лекарств или важные звонки, без необходимости осваивать сложные приложения.

Мой VK-бот разработан для широкого круга пользователей, которые ценят свое время и ищут простое, но эффективное решение для управления напоминаниями. Он идеально подходит для тех, кто ведет активный образ жизни.

4

Анализ и аналоги

Изученные VK-боты

«Напоминалка»: Простой бот для напоминаний.

«Нотификейк»: Бот с более продвинутыми функциями планирования.

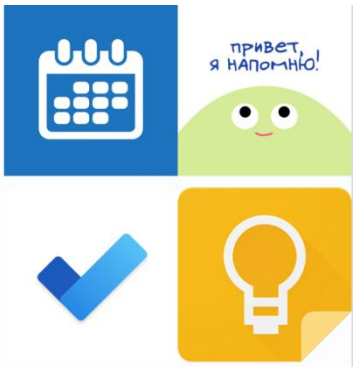
Приложения для заметок и задач

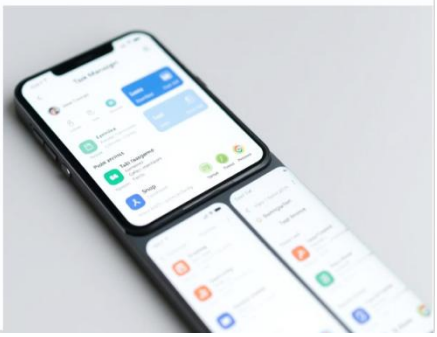
Google Keep: Облачное приложение для заметок и списков.

ToDoist: современное приложение для управления задачами и ежедневного планирования

Microsoft To Do: Приложение для управления задачами.

В ходе анализа были изучены как существующие VK-боты, так и популярные приложения для управления задачами и напоминаниями. Это позволило выявить сильные и слабые стороны каждого решения, а также определить ключевые функциональные возможности, которые будут реализованы в моем боте для обеспечения максимальной эффективности и удобства пользователей.





5

Функциональность бота

Добавление напоминаний

Пользователи могут добавлять напоминания

Управление напоминаниями

Полный контроль над созданными напоминаниями:

- Просмотр списка всех активных напоминаний.
- Удаление напоминаний, которые стали неактуальными.

Интерфейс и локализация

Удобный пользовательский опыт:

- Управление через текстовые команды и inline-кнопки.
- Полная поддержка русского языка.

6

Архитектура системы

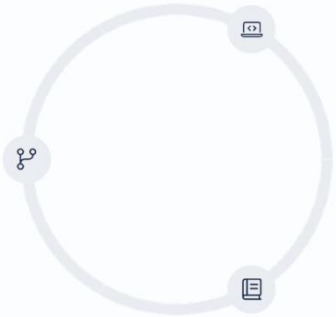
1	**Пользовательский интерфейс (VK UI)** Взаимодействие с пользователем через VK-клиент: команды, кнопки, сообщения.
2	**Бизнес-логика (NotificationManagerService)** Основной компонент, отвечающий за обработку запросов пользователей: добавление, удаление, модификация напоминаний, интеграция с Quartz.NET для планирования.
3	**Слой данных (NotificationStorageService)** Управление базой данных SQLite. Обеспечивает сохранение, чтение и удаление информации о напоминаниях, а также инициализацию структуры базы данных.
4	**Фоновый сервис (VkBotPollingService)** Основной сервис бота. Через Long Poll API постоянно получает сообщения от пользователей, управляет жизненным циклом приложения и обеспечивает взаимодействие между всеми слоями системы.

7

Используемые технологии

Язык: C# (.NET 8.0)

Выбран благодаря своей производительности, надежности и широкой поддержке библиотек для создания современных приложений.



IDE: Visual Studio 2022

Интегрированная среда разработки, предоставляющая мощные инструменты для кодирования, отладки и тестирования.

Библиотеки

- **VkNet**: основная библиотека для взаимодействия с VK API (Long Poll, отправка сообщений, работа с клавиатурами)
- **Quartz.NET**: для планирования и выполнения задач по расписанию.
- **Microsoft.Data.Sqlite + Entity Framework Core:** для работы с локальной базой данных SQLite (хранение напоминаний)

8

Основные компоненты

VkBotPollingService	Отвечает за постоянное прослушивание сообщений от пользователей через VK Long Poll API и их первичную обработку.
NotificationManagerService	Основная бизнес-логика приложения. Отвечает за добавление, просмотр, удаление напоминаний, парсинг даты и времени, а также интеграцию с планировщиком Quartz.NET.
NotificationStorageService	Обеспечивает взаимодействие с базой данных SQLite: создание таблиц, сохранение, получение и удаление информации о напоминаниях.
NotificationJob	Класс, реализующий логику отправки уведомлений пользователям по расписанию с использованием Quartz.NET.

9

Тестирование и интерфейс

Проведенные тестирования

Функциональное тестирование: Проверка соответствия бота заявленным функциям, корректности добавления, изменения и удаления напоминаний, а также их своевременной отправки.

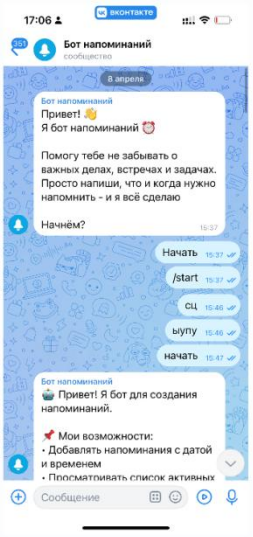
Интерфейсное тестирование: Оценка удобства использования интерфейса VK, интуитивности команд и работы inline-клавиатуры.

10

Тестирование и интерфейс

Начально окно приветствия:

- Экран приветствия
- Основные возможности бота
- Краткое руководство по использованию

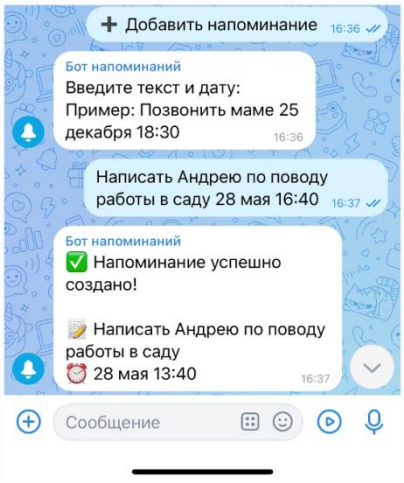


11

Тестирование и интерфейс

Добавление напоминаний:

- Создание напоминания
- Уведомление от телеграм-бота

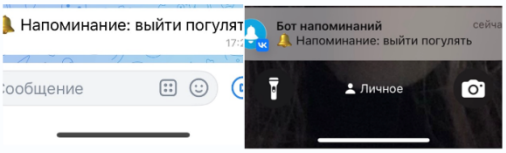


12

Тестирование и интерфейс

Уведомление о напоминании в диалоге и экране уведомлений:

- Бот успешно отправляет напоминания как в чат с пользователем, так и в виде push-уведомления на мобильном устройстве. Это позволяет пользователю не пропустить важное событие даже при закрытом приложении ВКонтакте.

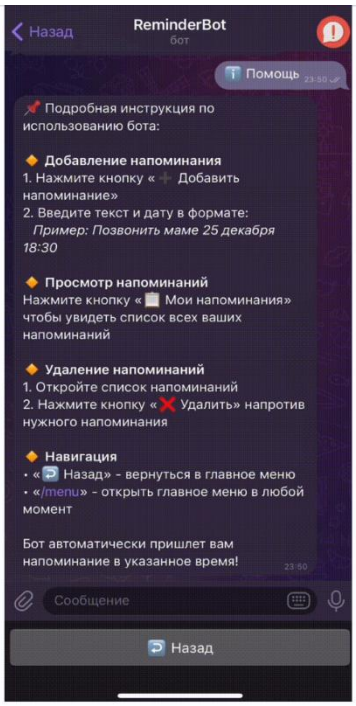


13

Тестирование и интерфейс

Подробная инструкция по использованию бота:

- Добавление напоминания
- Просмотр напоминаний
- Удаление напоминаний



14

Вывод по тестированию

Проведенные тестирования подтвердили высокую надежность и удобство использования ВК-бота «Напоминания». Интерфейс разработан таким образом, чтобы обеспечить максимальную простоту и ясность для любого пользователя, независимо от его опыта работы с мессенджерами и приложениями.

Возможность расширения:

- голосовые напоминания
- интеграция с календарями

https://vk.com/bot_napominaniv

15

Спасибо за внимание!