

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

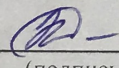
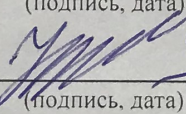
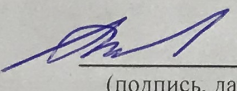
«Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»
Профиль подготовки: «Программное обеспечение средств вычислительной техники и автоматизированных систем»

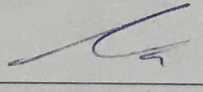
БАКАЛАВРСКАЯ РАБОТА

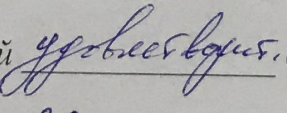
на тему

БОТ ДЛЯ СИСТЕМЫ DISCORD

Студент  31.05.2023 И.М. Кубасов
(подпись, дата)
Руководитель  31.05.2023 Н.В. Слепцов
(подпись, дата)
Нормоконтролёр  31.05.2023 А.С. Бычков
(подпись, дата)

Работа допущена к защите (протокол зас. кафедры от 02.06.23 г. № 14)

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

Работа защищена с отметкой  (протокол заседания ГЭК от
27.06.2023г. № 29)

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

Пенза 2023

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

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

«Утверждаю»

Заведующий кафедрой

М.А. Митрохин

« 26 » апреля 20 23 г.

**ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ**

1. Студент Кубасов Иван Михайлович гр 19ВВП1 факультета вычислительной техники.

2. Тема работы Бот для системы Discord

Тема утверждена приказом ПГУ № 458/о от "26" апреля 2023 г.

3. Руководитель бакалаврской работы к.т.н., доцент, каф. ВТ ПГУ Слепцов Н.В.

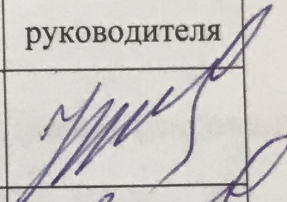
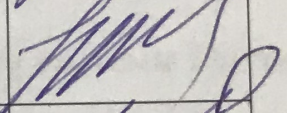
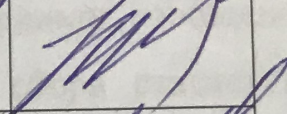
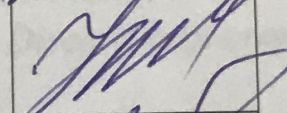
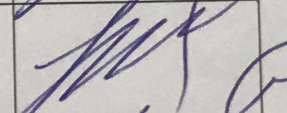
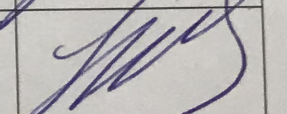
4. Задание на работу (назначение разработки, исходные данные и т.п.)

Реализовать бота для системы Discord, помогающего в администрировании сервера. Реализовать настраиваемый фильтр слов, систему поощрения и временную блокировку пользователей. Провести тестирование и отладку каждого элемента приложения.

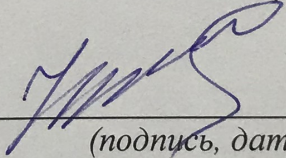
5. Перечень подлежащих разработке вопросов

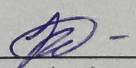
Требуется произвести анализ предметных областей: изучить существующих ботов, выделить их достоинства и недостатки. Выбрать язык программирования, который подойдет для написания логики приложения. Определить архитектуру приложения и вспомогательные средства разработки.

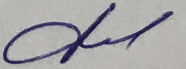
6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Изучение предметной области	50	26.04.2023-30.04.2023	
2.	Анализ требований	10	01.05.2023-02.05.2023	
3.	Выбор технологий, применяемых при проектировании и разработке	50	03.05.2023-07.05.2023	
4.	Описание алгоритмов	40	08.05.2023-11.05.2023	
5.	Реализация бота для системы Discord	150	12.05.2023-29.05.2023	
6.	Оформление пояснительной записки	24	30.05.2023-31.05.2023	

Дата выдачи задания "26" апреля 2023 г.

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

Задание к исполнению принял студент  Кубасов И.М.
(подпись, дата)

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

26.06.2023

Реферат

Бакалаврская работа содержит 81 лист, 30 рисунков, 1 таблицы, 5 источников и 3 приложения

БОТ, АДМИНИСТРИРОВАНИЕ, БАЗА ДАННЫХ, *DISCORD*, PYTHON, *SQL*, *POSTGRESQL*

Цель работы – реализация бота для *Discord*-сервера.

Объектом разработки является бот, облегчающий администрирование *Discord*-серверов.

В результате проделанной работы был реализован бот для системы *Discord*, облегчающий модерирование *Discord*-серверов. Разработан фильтр сообщений, разработана система поощрений пользователей и система сбора статистики. Предоставлены ценные инструменты для администрирования *Discord*-серверов и создания дружественной и благополучной среды общения.

					ПГУ1.090301.11.001 ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата	Бот для системы <i>Discord</i> Пояснительная записка	Лит.	Лист	Листов
Разраб		Кубасов И.М.					2	81
Провер.		Слепцов Н.В.						
Н. Контр		Бычков А.С.						
Утверд		Митрохин М.А.						
						ФВТ, каф. ВТ гр. 19ВВП1		

Содержание

Перечень сокращений и обозначений	4
Введение	5
1 Бот для системы Discord	6
1.1 Анализ существующих аналогов	7
1.1.1 JuniperBot	7
1.1.2 MEE6	8
1.1.3 ProBot	10
1.1.4 Dyno	11
1.2 Постановка задачи	12
1.3 Выбор языка программирования	13
1.3.1 Python	14
1.3.2 JavaScript	14
1.3.3 Ruby	14
1.3.4 Java	15
1.3.5 C#	15
1.4 Выводы	15
1.5 Выбор инструментов разработки	16
1.5.1 Выбор среды разработки	16
1.6 Архитектура приложения	21
1.7 Разработка приложения	22
1.7.2 Разработка модуля write_ban_thread.py	27
1.7.3 Разработка модуля lvl_inspector.py	27
1.7.4 Разработка модуля config.py	27
1.8 Тестирование и отладка	28
2 Охрана труда	39
2.1 Анализ негативных факторов, воздействующих на оператора ПЭВМ и способы их устранения	39
2.1.1 Оборудование рабочего места	39

2.1.2 Освещенность рабочего места	41
2.1.3 Электробезопасность	42
2.1.4 Пожарная безопасность	42
2.1.5 Режим труда и отдыха	43
Заключение.....	45
Список использованных источников	46
Приложение А – UML – диаграммы	47
Приложение Б – Листинг программы	52
Приложение В – Презентация.....	64

					ПГУ1.090301.11.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		3

Перечень сокращений и обозначений

В настоящей работе применяют следующие сокращения и обозначения

Акроним – разновидность аббревиатуры, которая образуется по первым буквам слов, входящих в зашифрованное понятие

Бот – специальная программа, выполняющая какие-либо действия через интерфейсы, предназначенные для пользователей

ГОСТ – региональный стандарт, принятый Межгосударственным советом по стандартизации, метрологии и сертификации Содружества Независимых Государств

Геймер – человек, играющий в видеоигры

Киберспортсмен – человек, играющий в видеоигры и выступающий на соревнованиях, где получает за успешные выступления вознаграждения

Лемматизация – приведение слова к своей базовой форме

Модерация – контроль над соблюдением требований какого-либо сообщества

Плагин – независимо компилируемый программный модуль, подключаемый к программе и предназначенный для расширения ее возможностей

ПК – персональный компьютер

ПЭВМ – персональная электронно-вычислительная машина

Framework – программная платформа, которая упрощает разработку программного продукта, определяет структуру проекта

IDE (Integrated Development Environment) – интегрированная среда разработки

.NET – кроссплатформенная платформа для создания различных типов приложений

Nickname – имя пользователя, псевдоним

UML (Unified Modeling Language) – унифицированный язык моделирования

					ПГУ1.090301.11.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		4

Введение

Discord — одно из лучших приложений-мессенджеров, которое отлично справляется со всеми возможными задачами по общению в сети, включая пересылку файлов, текстовые диалоги и видеозвонки, и при этом имеет низкие системные требования. Киберспортсмены получают от этого больше всего преимуществ, поскольку свободные аппаратные ресурсы вроде мощности процессора и оперативной памяти позволяют не терять производительность своей системы и добиться максимально красочного и стабильного изображения в требовательных играх даже во время звонков. Кроме того, поддерживается функция наложения изображения, которая выводит на экран поверх картинки из игры *nickname* всех, кто находится в голосовом канале, а когда кто-то из собеседников говорит, его имя подсвечивается.

Сегодня *Discord* используют не только геймеры, его используют большие компании и образовательные программы. С каждым днем пользователей и серверов становится все больше, крупные компании приходят и создают свои сообщества

Целью выпускной работы является разработка и реализация бота для *Discord*-сервера, который будет обладать различными функциями, направленными на упрощение процесса администрирования и обеспечение удобства пользователям.

					ПГУ1.090301.11.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		5

1 Бот для системы Discord

Системы общения и взаимодействия между пользователями на платформе *Discord* стали невероятно популярными в последние годы. Они создают коммуникационное пространство, где пользователи могут собираться, обмениваться информацией и взаимодействовать друг с другом. Однако, с ростом популярности *Discord*-серверов возникают потребности в эффективном администрировании и управлении ими.

Важность данной работы заключается в нескольких аспектах:

- развитие сообществ и коммуникации. Бот для системы *Discord* с описанными функциями способствует развитию сообществ и обеспечивает эффективную коммуникацию между участниками. Он предоставляет удобные инструменты для управления сервером, поддерживает активность пользователей и стимулирует их участие, что способствует созданию взаимопонимания и сотрудничества;

- улучшение администрирования. Бот, осуществляющий функции администрирования, помогает облегчить работу администраторов сервера. Он автоматизирует рутинные задачи, предоставляет инструменты для контроля активности пользователей и обеспечивает безопасность и порядок на сервере;

- мониторинг активности и аналитика. Функции подсчета времени, отправленных сообщений и общего времени на сервере позволяют получить ценную информацию об активности пользователей и поведении сообщества в целом. Это помогает администраторам принимать информированные решения, разрабатывать стратегии развития сервера и улучшать взаимодействие с участниками;

- создание приятной и безопасной атмосферы. Фильтр слов и система выдачи ролей на основе активности помогают поддерживать приятную и безопасную атмосферу на сервере. С помощью разрабатываемого бота будет возможным контролировать нежелательную лексику и поведение пользователей, что поспособствует созданию дружелюбной и уважительной среды для всех участников;

					ПГУ1.090301.11.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		6

- улучшение управления и эффективности. Бот значительно оптимизирует управление сервером и обеспечивает эффективное использование ресурсов. Администраторы могут сосредоточиться на более важных задачах и взаимодействии с участниками, в то время как бот автоматически выполняет рутинные задачи и предоставляет необходимую информацию.

В целом, разработка и реализация бота для системы *Discord* с описанными функциями имеют важное значение для создания активного, безопасного и хорошо управляемого сообщества, где участники могут эффективно общаться и сотрудничать.

1.1 Анализ существующих аналогов

Среди аналогов *Discord* ботов можно выделить:

- *JuniperBot*;
- *MEE6*;
- *ProBot*;
- *Dyno*.

1.1.1 *JuniperBot*

JuniperBot является многофункциональным ботом для *Discord*. Он предлагает различные полезные функции, такие как модерация чата, приветствие новых пользователей, музыкальные команды и т. д. *JuniperBot* имеет настраиваемые параметры, что позволяет администраторам сообщества настраивать его под свои нужды.

Профиль приложения *JuniperBot* представлен на рисунке 1.

Ссылка на официальный сайт - <https://juniper.bot>.

Используя этого бота в течение нескольких дней, можно выделить следующие плюсы:

- многофункциональность;
- настраиваемые параметры;
- поддержка музыкальных команд.

Из минусов можно выделить следующее:

					ПГУ1.090301.11.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		7

- ограниченная поддержка;
- ограниченные интеграции;
- сложность настройки.

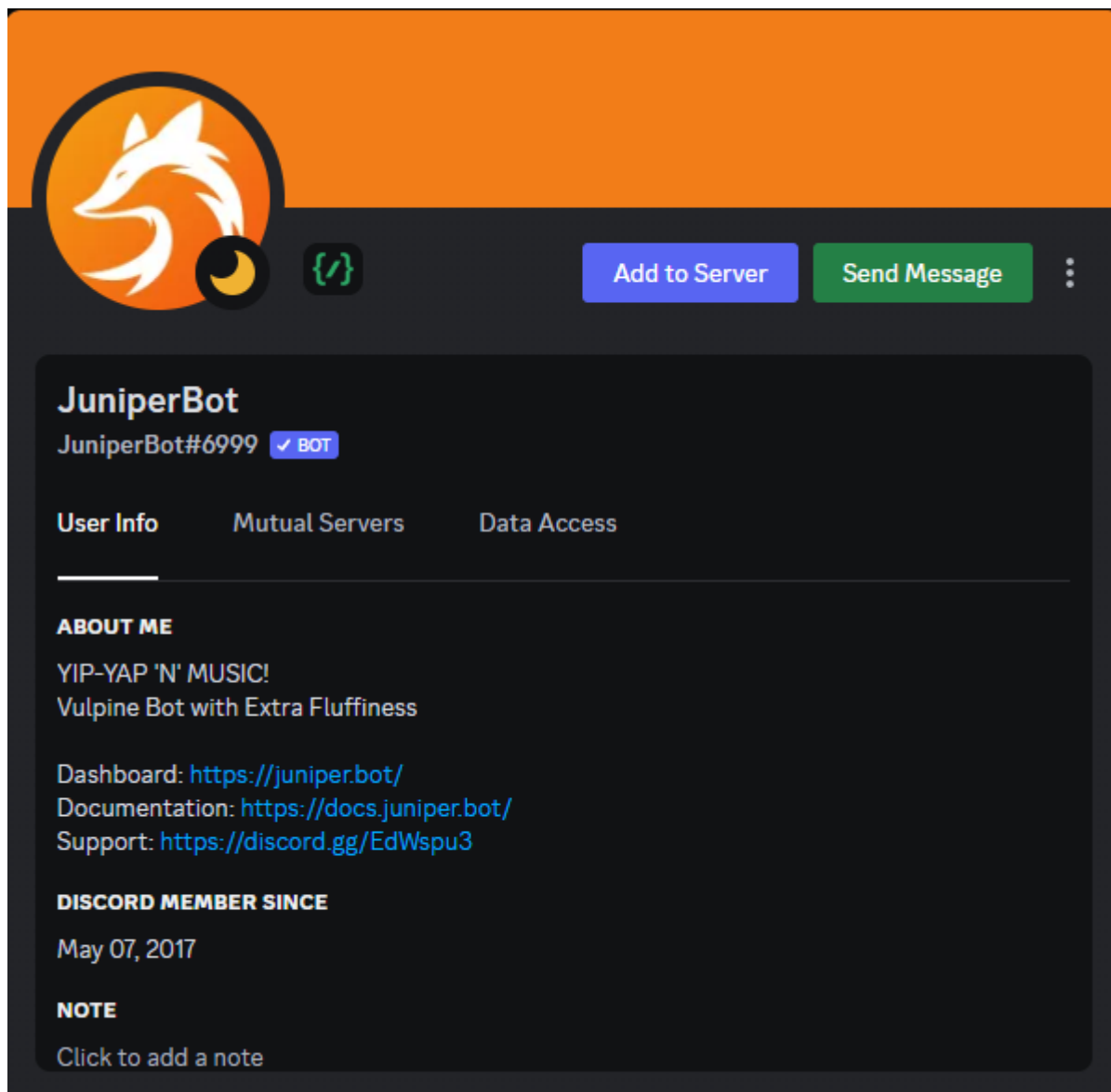


Рисунок 1 – Профиль приложения *JuniperBot*

1.1.2 MEE6

MEE6 также является многофункциональным ботом для *Discord*. Он предоставляет функции уровней и ролей, системы наград, модерации чата, настраиваемых приветствий и прощаний, а также музыкальные команды. *MEE6* позволяет администраторам настраивать роли и команды, а также создавать автоматические роли для пользователей на основе активности.

					ПГУ1.090301.11.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		8

Профиль приложения *MEE6* представлен на рисунке 2.

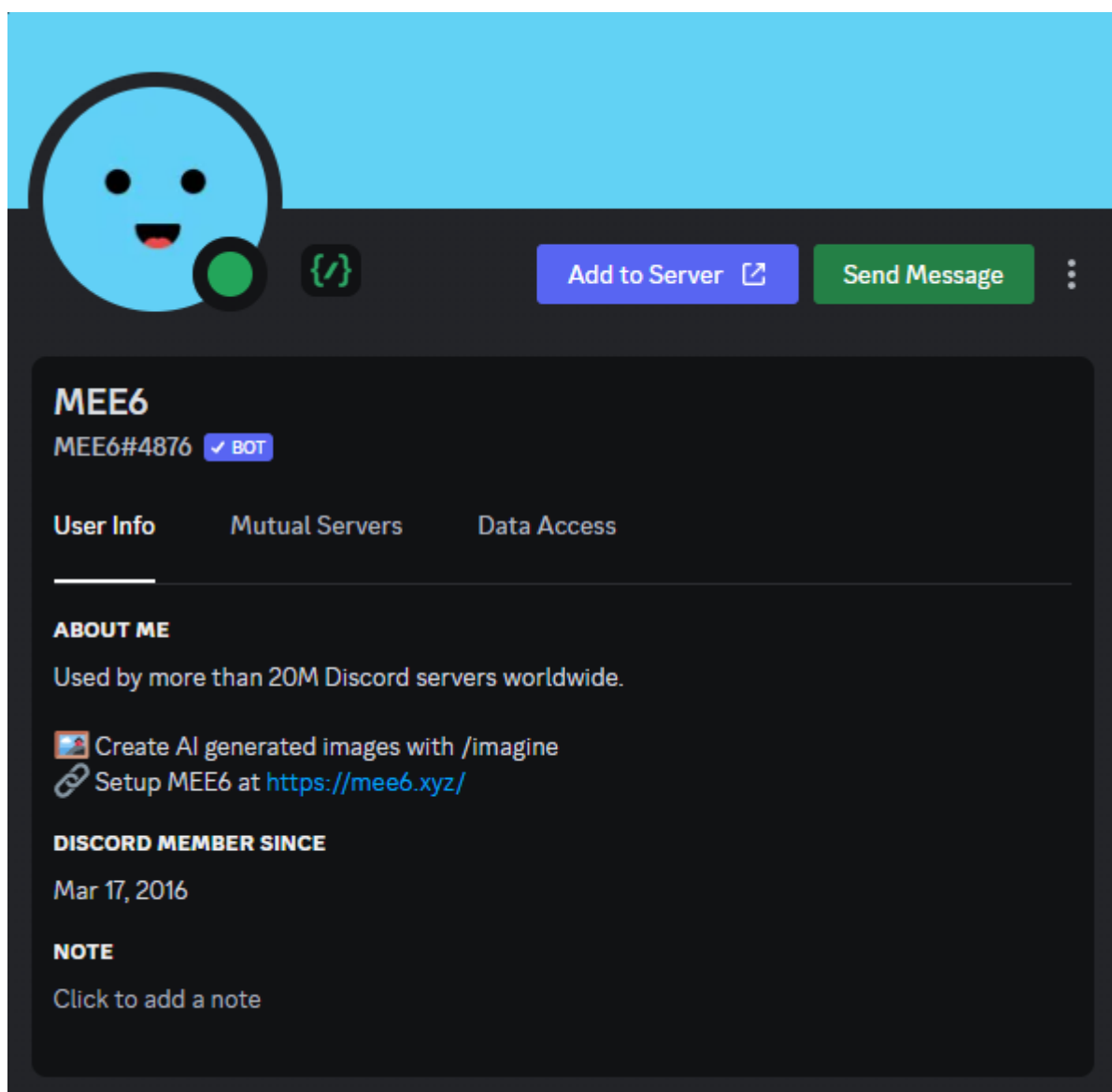


Рисунок 2 – Профиль приложения *MEE6*

Ссылка на официальный сайт приложения - <https://mee6.xyz/en>.

Используя этого бота в течение нескольких дней, можно выделить следующие плюсы:

- функции уровней и ролей;
- настраиваемые команды;
- система наград.

Из минусов можно выделить следующее:

- ограниченный бесплатный функционал;

					ПГУ1.090301.11.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		9

- ограниченные настройки и гибкость;
- ограниченная интеграция с другими сервисами;
- низкая скорость ответа и ограниченная поддержка;
- ограниченная настраиваемая команда.

1.1.3 ProBot

ProBot является ещё одним многофункциональным ботом для *Discord*. Он предлагает различные функции, включая настраиваемые приветствия, автомодерацию, модерацию чата, настраиваемые команды и т. д. *ProBot* также поддерживает плагины, что позволяет расширять его функциональность и добавлять новые возможности.

Профиль приложения *ProBot* представлен на рисунке 3.

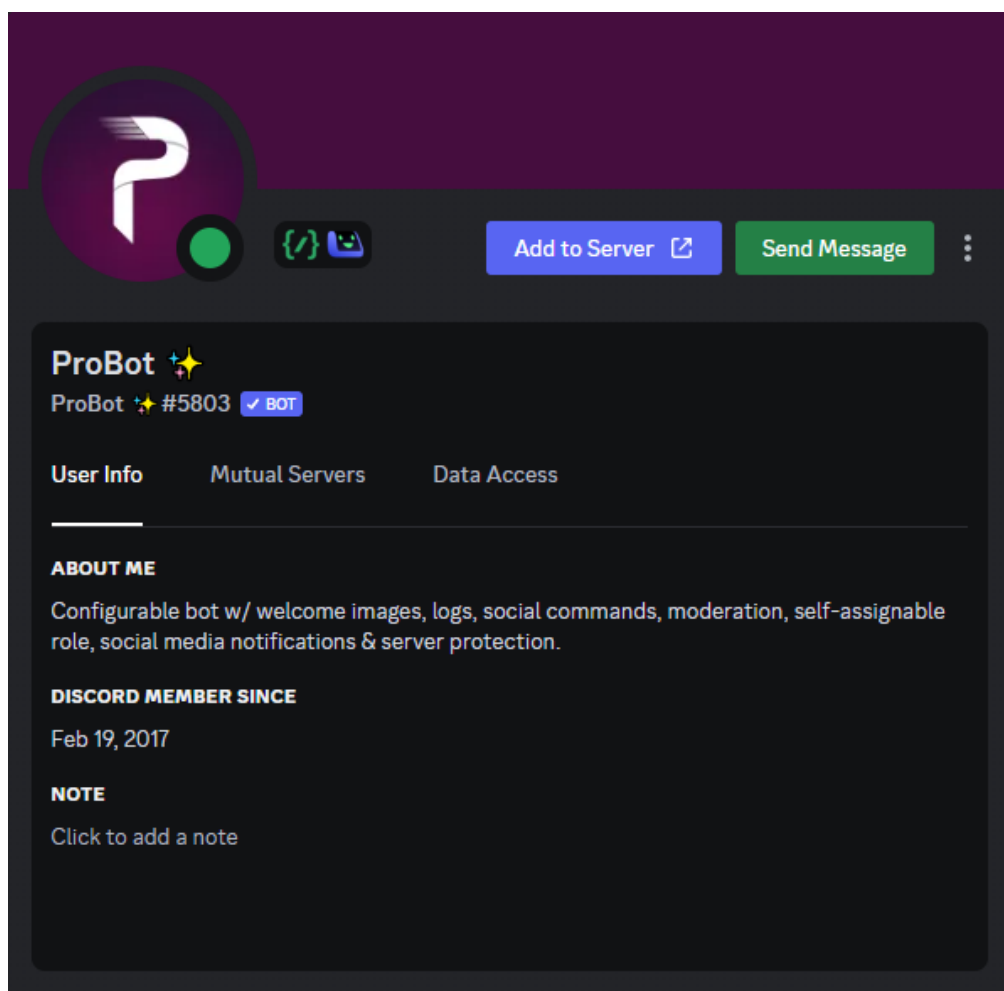


Рисунок 3 – Профиль приложения *ProBot*

<https://top.gg/bot/282859044593598464>.

Используя этого бота в течение нескольких дней, можно выделить следующие плюсы:

- настраиваемые приветствия;
- модерация чата;
- возможность расширения через плагины.

Из минусов можно выделить следующее:

- ограниченный набор функций и возможностей;
- сложность настройки;
- недостаток графического интерфейса;
- ограниченная поддержка и документация;
- требуется самостоятельное развертывание.

1.1.4 Dyno

Dyno - это многофункциональный бот для управления сервером *Discord*. Он предлагает различные функции администрирования и модерации, которые помогают в управлении сообществом.

Профиль приложения *Dyno* представлен на рисунке 4.

Используя этого бота в течение нескольких дней, можно выделить следующие плюсы:

- множество инструментов для модерации сервера;
- позволяет создавать команды, которые могут выполнять различные задачи;
- возможность настраивать автоматическое присвоение ролей новым участникам сервера;
- возможность настройки автоматического отправления приветственных сообщений новым участникам сервера.

Из минусов можно выделить следующее:

- ограниченные бесплатные функции;
- сложность настройки;

- возможные проблемы и задержки.

Ссылка на официальный сайт - <https://dyno.gg>.

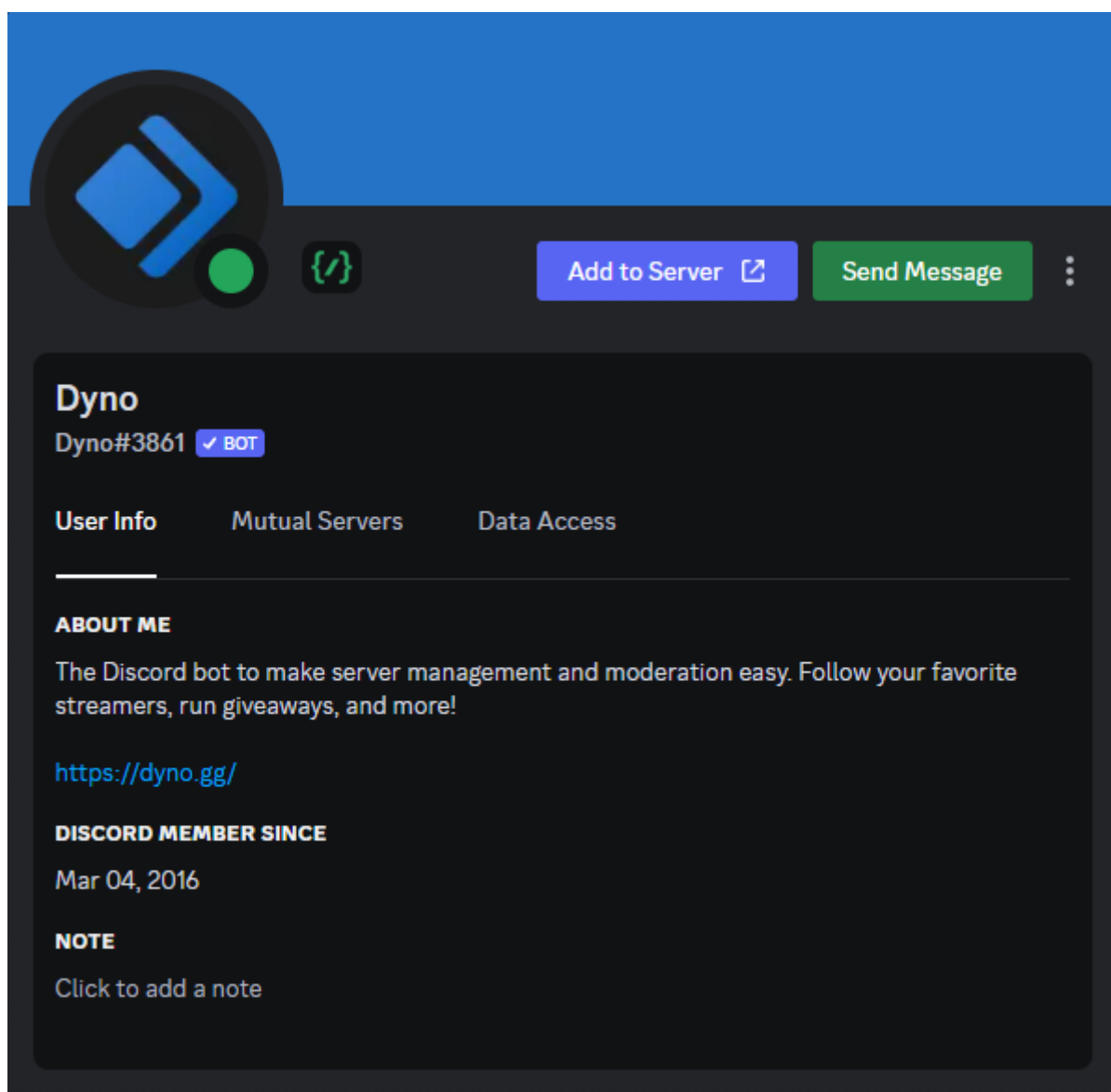


Рисунок 4 – Профиль приложения *Dyno*

1.2 Постановка задачи

Цель данной дипломной работы состоит в разработке и реализации бота для системы *Discord*, который будет обладать следующими функциями:

- администрирование. Бот должен иметь функции управления и администрирования сервера. Возможности администрирования включают выдачу и снятие ролей, отключение пользователей;

					ПГУ1.090301.11.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		12

- подсчёт времени в голосовых каналах. Бот должен отслеживать активность пользователей в голосовых каналах и вознаграждать их;
- подсчёт отправленных сообщений. Бот должен отслеживать количество отправленных сообщений каждым пользователем на сервере и вознаграждать их за активность;
- выдача "очков опыта" и ролей. Бот должен иметь систему "очков опыта", которая будет награждать пользователей за их активность на сервере. Бот должен предоставлять возможность выдачи ролей пользователям на основе накопленных "очков опыта". Роли должны расширять возможности пользователей в соответствии с их достижениями и активностью;
- выдача статистики. Бот должен давать возможность пользователю узнать количество накопленных им "очков опыта";
- временная блокировка. Бот должен иметь функционал для временной блокировки (бана) пользователей на сервере. Администраторы должны иметь возможность установить временный бан для определенного пользователя и автоматически разблокировать его по истечении заданного времени;
- настраиваемый фильтр нецензурной лексики. Бот должен обладать функционалом фильтра слов, который будет контролировать отправляемые сообщения и блокировать использование запрещенных или нежелательных слов. Бот должен предоставлять настройки для добавления и изменения списка запрещенных слов.

Поставленные задачи включают в себя проектирование архитектуры бота, разработку логики функций, взаимодействие с *API Discord*, реализацию системы хранения данных (например, базы данных), развертывание бота на сервере *Discord* и тестирование его функциональности.

1.3 Выбор языка программирования

Реализацию бота для *Discord* можно осуществить с использованием различных языков программирования и технологий.

					ПГУ1.090301.11.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		13

1.3.1 Python

Python является популярным языком программирования для разработки ботов для *Discord*. Он имеет простой и понятный синтаксис, обширную библиотеку *discord.py* для взаимодействия с *API Discord*, и большое сообщество разработчиков, что обеспечивает поддержку и ресурсы для разработчиков. *Python* также обладает хорошей производительностью и подходит для быстрой разработки прототипов.

Однако, стоит отметить, что возможные минусы включают некоторое снижение производительности по сравнению с некоторыми другими языками, особенно при работе с высоконагруженными системами.

1.3.2 JavaScript

JavaScript широко используется для разработки ботов для *Discord*. Он является основным языком программирования для веб-разработки, и многие разработчики уже имеют опыт работы с *JavaScript*, что облегчает изучение и использование для разработки ботов. *Discord* предоставляет официальную библиотеку *Discord.js* для взаимодействия с *API Discord* на основе *JavaScript*.

Однако, *JavaScript* может иметь несколько сложный синтаксис и некоторые особенности, которые могут вызвать путаницу для новичков. Также, при работе с большими проектами, *JavaScript* может столкнуться с проблемами масштабируемости и производительности.

1.3.3 Ruby

Ruby является простым и читаемым языком программирования, что делает его привлекательным для разработки ботов для *Discord*. Библиотека *Discordrb* предоставляет удобные инструменты для взаимодействия с *API Discord* на языке *Ruby*. *Ruby* также имеет активное сообщество разработчиков.

Однако, *Ruby* не так популярен, как *Python* или *JavaScript*, поэтому возможно найти меньше ресурсов и документации по разработке ботов для *Discord* на *Ruby*. Также, некоторые разработчики могут не быть знакомыми с *Ruby* и потребуется время на его изучение.

					ПГУ1.090301.11.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		14

1.3.4 Java

Java является мощным языком программирования с широким применением и большим сообществом разработчиков. Существуют библиотеки, такие как *JDA* (*Java Discord API*), которые предоставляют функциональность для разработки ботов для *Discord* на *Java*. *Java* обладает высокой производительностью и хорошей масштабируемостью.

Однако, стоит отметить, что *Java* может иметь более сложный синтаксис и требовать более многословного кода по сравнению с некоторыми другими языками. Он также может потребовать больше ресурсов для развертывания и запуска бота.

1.3.5 C#

C# - мощный и многофункциональный язык программирования, который может быть использован для разработки ботов для *Discord*. Библиотека *Discord.Net* предоставляет возможности для создания *Discord*-ботов на основе *C#*. *C#* имеет хорошую производительность и широкие возможности для разработки сложных приложений.

Однако, *C#* может иметь более высокий порог входа для начинающих разработчиков и требовать более глубокого понимания объектно-ориентированного программирования. Также, по сравнению с некоторыми другими языками, *C#* может иметь более ограниченное сообщество разработчиков для *Discord*.

1.4 Выводы

Каждый язык программирования имеет свои преимущества и недостатки, и выбор языка зависит от конкретных задач, которые необходимо решить. Однако для реализации бота был выбран язык программирования *Python*, так как он предоставляет богатые возможности для разработки ботов для *Discord* и имеет популярную библиотеку *discord.py*, которая облегчает взаимодействие с *API Discord*.

					ПГУ1.090301.11.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		15

1.5 Выбор инструментов разработки

1.5.1 Выбор среды разработки

Существует множество интегрированных сред разработки (*IDE*) для программирования на *Python*. Некоторые из наиболее популярных *IDE*, которые поддерживают *Python*, это *Microsoft Visual Studio*, *PyCharm*, *Wing*.

Visual Studio — это интегрированная среда разработки, предназначенная для разработки программного обеспечения под различные платформы, включая *Windows*, *Android*, *iOS* и многие другие. Она была впервые выпущена в 1997 году и предназначалась для разработки приложений под операционную систему *Windows NT 4.0*. В то время она предлагала инструменты для работы с языками программирования, такими как *Visual Basic*, *C++* и *Visual J++*.

Со временем *Visual Studio* получила множество обновлений и значительно расширила свой функционал. В 2002 году вышла версия *Visual Studio .NET*, которая добавила поддержку языка *C#* и платформы *.NET Framework*. В 2005 году была выпущена *Visual Studio 2005*, в которой добавились возможности разработки веб-приложений и инструменты для создания мобильных приложений.

На сегодняшний день *Visual Studio* является одной из самых популярных *IDE* для разработки программного обеспечения, которая поддерживает множество языков программирования, включая *C++*, *C#*, *F#*, *Visual Basic*, *JavaScript*, *TypeScript*, *Python* и многие другие.

Среди преимуществ *Visual Studio* можно выделить следующие:

- поддержка *Unity*;
- наличие бесплатной *Community*-версии;
- богатый функционал и множество инструментов для разработки, отладки и тестирования кода;
- высокая интеграция с другими продуктами *Microsoft*, такими как *Azure* и *Git*;
- поддержка широкого спектра языков программирования.

					ПГУ1.090301.11.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		16

Однако есть и некоторые недостатки:

- требуется значительное количество ресурсов для запуска и использования, что может быть проблемой для менее мощных компьютеров;
- платная версия может быть слишком дорогой для некоторых пользователей.

Wing IDE — это популярная среда разработки программного обеспечения, предназначенная для разработчиков на языке *Python*. Она разработана компанией *Wingware* и предлагает множество функций и инструментов для облегчения процесса разработки, от отладки кода до управления проектами.

Wing IDE обладает интуитивно понятным пользовательским интерфейсом, который обеспечивает удобное и продуктивное программирование. Он предлагает различные раскраски синтаксиса и настраиваемые темы оформления, чтобы удовлетворить предпочтения каждого разработчика.

Одной из ключевых особенностей *Wing IDE* является его мощный отладчик. Он позволяет разработчикам выполнять пошаговую отладку, наблюдать значения переменных, следить за стеком вызовов и точками останова в своем коде. Отладчик *Wing IDE* интегрирован в среду разработки и предоставляет удобный интерфейс для анализа и исправления ошибок.

Кроме того, *Wing IDE* предлагает автоматическое завершение кода, что упрощает написание кода и повышает производительность. Он также обеспечивает инструменты для анализа кода, включая статическое анализирование, проверку типов и подсказки по *API*. Это помогает разработчикам обнаруживать потенциальные ошибки и улучшать качество своего кода.

Среда разработки *Wing IDE* также предоставляет возможности для управления проектами, включая интеграцию с системами контроля версий, такими как *Git*. Она поддерживает работу с различными фреймворками и библиотеками *Python*, обеспечивая инструменты для создания, отладки и развертывания приложений.

В целом, *Wing IDE* представляет собой мощное средство разработки *Python*, которое помогает программистам увеличить их производительность и

					ПГУ1.090301.11.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		17

эффективность при разработке программного обеспечения. Она объединяет интуитивно понятный интерфейс с богатым набором функций и инструментов, делая процесс разработки более удобным и эффективным.

Среди преимуществ *Wing IDE* можно выделить следующие:

- легковесность. *Wing* является относительно легкой и быстрой *IDE*, что позволяет ей работать плавно даже на старых или менее мощных компьютерах;
- мощный отладчик. *Wing* предлагает мощные возможности отладки, включая точки останова, пошаговое выполнение кода, просмотр значений переменных и многое другое;
- хорошая документация. *Wing* имеет хорошо организованную и подробную документацию, что делает его привлекательным для новичков, которым нужна помощь в изучении *Python* и использовании *IDE*.

Однако есть и некоторые недостатки:

- ограниченные функции. В сравнении с *PyCharm*, *Wing* имеет меньший набор функций и инструментов. Некоторые разработчики могут потребовать дополнительных плагинов или расширений, чтобы удовлетворить свои потребности.
- ограниченная интеграция. В отличие от *PyCharm*, интеграция *Wing* с другими инструментами и фреймворками может быть ограниченной или менее прямолинейной.

Выбор между *PyCharm* и *Wing* в значительной степени зависит от личных предпочтений и потребностей разработчика. Если вы ищете мощную и полнофункциональную *IDE* с широкой поддержкой фреймворков и инструментов, *PyCharm* может быть хорошим выбором. С другой стороны, если вы предпочитаете более легковесную и быструю *IDE* с сильным акцентом на отладке, *Wing* может быть более подходящим вариантом.

PyCharm - это интегрированная среда разработки для языка программирования *Python*, созданная компанией *JetBrains*. Она предоставляет мощные инструменты и функциональность, которые помогают разработчикам повысить эффективность и продуктивность при работе над проектами на *Python*.

					ПГУ1.090301.11.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		18

PyCharm обладает широким набором функций, предназначенных для упрощения разработки *Python*-приложений. Вот некоторые из основных возможностей:

- интеллектуальный редактор кода. *PyCharm* обладает мощным редактором кода, который предлагает автодополнение, подсветку синтаксиса, проверку ошибок и другие функции, чтобы сделать процесс написания кода более эффективным;

- отладка. *PyCharm* предоставляет полнофункциональный отладчик, позволяющий разработчикам искать и исправлять ошибки в своем коде. Он поддерживает шаг за шагом выполнение кода, точки останова, просмотр переменных и многое другое;

- управление проектами. *PyCharm* позволяет легко создавать и управлять проектами *Python*. Он предлагает функции, такие как система управления версиями (например, *Git*), интеграция с виртуальными средами *Python*, управление зависимостями и другие инструменты для эффективной организации проекта;

- инструменты для тестирования. *PyCharm* имеет встроенную поддержку модульного и функционального тестирования. Разработчики могут легко создавать и запускать тесты, а также анализировать покрытие кода, чтобы убедиться в его качестве;

- интеграция с другими инструментами. *PyCharm* интегрируется с широким спектром инструментов и плагинов, которые облегчают разработку на *Python*. Например, он поддерживает интеграцию с системами управления базами данных, веб-фреймворками, средствами анализа кода и другими инструментами;

- расширяемость. *PyCharm* имеет открытую архитектуру, которая позволяет разработчикам создавать собственные плагины и расширения. Это делает возможным настройку и расширение функциональности IDE под конкретные потребности разработчика.

					ПГУ1.090301.11.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		19

В целом, *PyCharm* является мощным инструментом для разработки на *Python*, предлагая широкий спектр функций, интеграцию с другими инструментами и удобную среду для создания проектов *Python* любого масштаба и сложности.

Среди преимуществ *PyCharm* можно выделить следующие:

- широкий набор функций. *PyCharm* предлагает широкий спектр инструментов для автоматического завершения кода, анализа кода, рефакторинга, контроля версий и многих других функций, которые облегчают разработку;
- удобство в использовании. Интерфейс *PyCharm* интуитивно понятен и легок в использовании, особенно для тех, кто привык работать в среде *IntelliJ IDEA*;
- богатая экосистема. *PyCharm* интегрируется с различными инструментами и фреймворками, такими как *Django*, *Flask*, *SQLAlchemy*, *Docker*, *Git* и многими другими;
- поддержка множества платформ. *PyCharm* доступен для *Windows*, *macOS* и *Linux*, что делает его универсальным выбором для разработчиков, работающих на разных операционных системах.

Однако есть и некоторые недостатки:

- ресурсоемкость. *PyCharm* является мощным инструментом, но при этом потребляет больше ресурсов компьютера в сравнении с некоторыми другими IDE, что может привести к некоторому снижению производительности на старых или менее мощных компьютерах;
- стоимость. *PyCharm* предлагает платную версию, но также имеет бесплатную "*Community Edition*" с ограниченным набором функций. Для полноценного использования всех возможностей *PyCharm* требуется приобретение лицензии.

					ПГУ1.090301.11.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		20

1.6 Архитектура приложения

В начале проектирования была составлена диаграмма развертывания для визуализации взаимодействия различных устройств на физическом уровне, она представлена на рисунке 5.

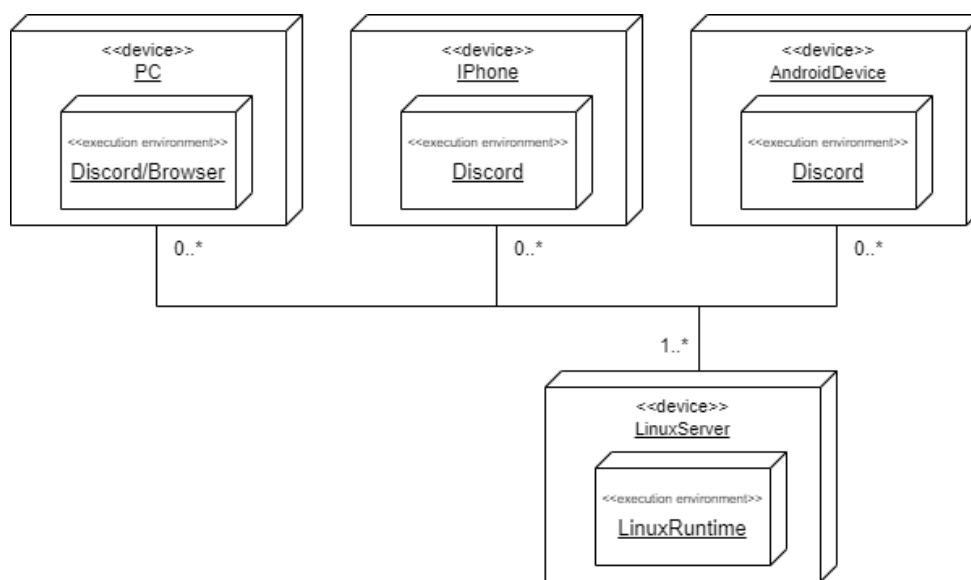


Рисунок 5 – UML-диаграмма развертывания

Архитектура бота для администрирования *Discord*-серверов, состоящего из нескольких модулей, выглядит следующим образом.

Config.py. Этот модуль содержит конфигурационные данные для подключения к базе данных (БД) и *Discord API*. Включает в себя переменные, такие как *data_base*, *user*, *password*, *host*, *port*, и *TOKEN*.

Lvl_inspector.py. В этом модуле определена функция *check_lvl*, которая проверяет уровень пользователя на основе его активности и прогресса. Она используется для автоматического назначения ролей в зависимости от достижения определенного уровня.

Write_ban_thread.py. В этом модуле определена асинхронная функция *write_ban_sleep*, которая используется для временного запрета пользователя на

отправку сообщений в чате. Она выполняет ожидание заданного времени и затем удаляет ограничения на отправку сообщений у пользователя.

Main.py. Это главный модуль, который связывает все остальные модули и реализует логику работы бота.

Импортируются необходимые зависимости, такие как *discord*, *discord.ext.commands*, *datetime*, *psycopg2* и *pymorphy2*, загружаются конфигурационные данные из *config.py*. Устанавливается соединение с базой

данных *PostgreSQL*. Создаются таблицы *ban_words* и *users*, если они еще не существуют. Закрывается соединение с базой данных. Создается экземпляр бота с помощью *commands.Bot* из *discord.ext*. Определяются команды бота *remove_ban_word*, *add_ban_word* и *write_ban*. Определяются обработчики событий *on_ready*, *on_message*, *on_voice_state_update* и *on_member_join*. Бот запускается с использованием токена из конфигурационных данных.

Общая архитектура включает подключение к базе данных, обработку команд и событий, выполнение функций проверки уровня и временного запрета отправки сообщений. Бот реагирует на различные события на сервере и выполняет соответствующие действия, такие как проверка запрещенных слов, назначение ролей, подсчет активности пользователя и другие административные функции.

1.7 Разработка приложения

Составим *UML*-диаграмму прецедентов для того, чтобы продемонстрировать, как акторы будут взаимодействовать с системой. Прецедент – это конкретный сценарий использования системы. Поскольку нельзя рассматривать разрабатываемого в рамках данной работы бота в отрыве от *Discord*-сервера, на который он будет добавлен, системой в данном случае будем считать *Discord*-сервер с установленным ботом. Акторами, взаимодействующими с системой, в данном случае будут пользователи и администраторы. *UML*-диаграмма изображена на рисунке 6.

					ПГУ1.090301.11.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		22

1.7.1 Разработка модуля main.py

В данном модуле реализованы все основные функции бота. Здесь определены обработчики сообщений и команд, поступающих от пользователей и администраторов, импортированы необходимые библиотеки, подключена база данных и создан экземпляр бота.

Одной из основных функций бота является проверка сообщений пользователей на употребление запрещённых слов, список которых определяется администратором. При этом могут возникнуть следующие проблемы: в случае использования строгого сравнения слов, администратору придётся перечислять запрещённые слова во всех возможных формах, что, учитывая их количество в русском языке, займёт очень много времени.

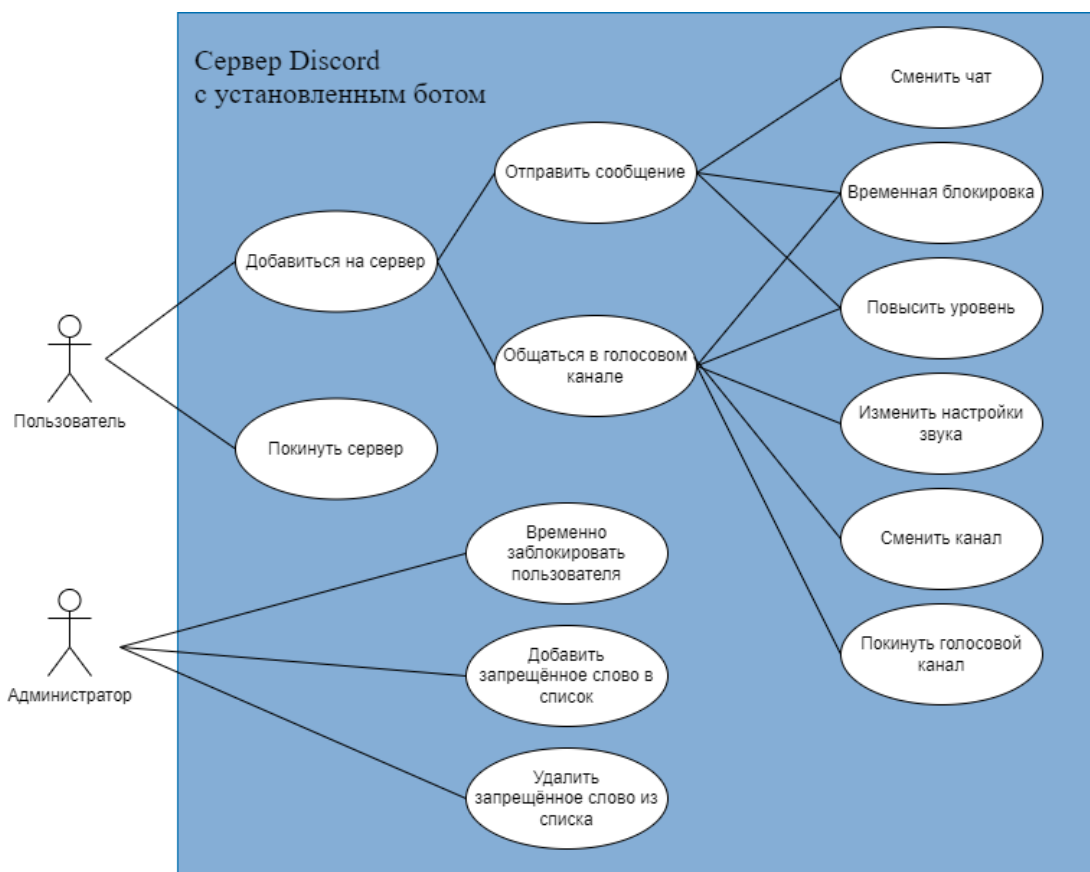


Рисунок 6 – UML-диаграмма прецедентов

Если же перечислить только корни слов, и проверять вхождение их в слово как подстроки, повышается шанс ложных срабатываний, когда корень запрещённого слова совпадает с частью другого слова, которая не является

корнем. Чтобы решить эту проблему и облегчить администратору работу, подойдёт библиотека *Rymorhy2*.

Rymorhy2 — это библиотека для морфологического анализа слов русского языка. При работе используется словарь *OpenCorpora*; для незнакомых слов строятся гипотезы. Также имеется экспериментальный словарь украинского языка. Она предоставляет программистам возможность лемматизировать и склонять слова, а также получать информацию о грамматических характеристиках слова, таких как часть речи, род, падеж и других.

Предназначение библиотеки *Rymorhy2* состоит в обработке текста на русском языке для решения различных задач, связанных с анализом и обработкой естественного языка. Она может использоваться в различных приложениях, таких как построение поисковых систем, автоматическое аннотирование текстов, машинный перевод и других задачах, требующих работать с естественным языком.

Основные функции библиотеки *Rymorhy2* включают:

- лемматизация. *Rymorhy2* позволяет приводить слова к их базовой форме (лемме). Например, для слова "бежал" леммой будет "бежать". Это полезно, когда необходимо обрабатывать разные формы одного слова как единое целое;
- склонение. Библиотека позволяет склонять слова в различные падежи, числа, роды. Например, можно просклонять слово "стол" в родительный падеж и единственное число, чтобы получить "стола";
- получение грамматических характеристик. *Rymorhy2* предоставляет информацию о части речи слова, его роде, падеже, числе и других грамматических характеристиках. Это может быть полезно при анализе текста и построении правил для обработки естественного языка.

Преимущества библиотеки *Rymorhy2*:

- простота использования. *Rymorhy2* имеет простой и понятный интерфейс, что делает его легким в освоении и использовании;

					ПГУ1.090301.11.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		24

- высокая производительность. Библиотека оптимизирована для быстрой обработки текста на русском языке, что позволяет эффективно использовать ее в реальных приложениях. Заявленное быстродействие составляет от нескольких тысяч слов в секунду до 100 тысяч слов в секунду в зависимости от выполняемой операции, интерпретатора и установленных пакетов. Потребление памяти – до 20 Мб;

- широкий охват грамматических характеристик. *Rymorphy2* предоставляет обширную информацию о грамматических характеристиках слов, позволяя более глубоко анализировать их семантическое значение и контекст.

Недостатки библиотеки *Rymorphy2*:

ограниченная поддержка языков. *Rymorphy2* разработана специально для работы с русским языком, имеет экспериментальную поддержку украинского языка и не поддерживает другие языки. Если нужно обрабатывать тексты на других языках, потребуется использовать другие библиотеки;

- не всегда точная лемматизация. Как и любой инструмент для морфологического анализа, *Rymorphy2* может допускать ошибки при лемматизации сложных словоформ или редких слов. Это может потребовать дополнительных проверок и обработки результатов;

- необходимость предварительной установки и загрузки данных. Чтобы начать использовать *Rymorphy2*, вам потребуется установить библиотеку и загрузить словари и модели, которые используются для морфологического анализа. Это может занять некоторое время и потребовать дополнительного хранилища.

В целом, библиотека *Rymorphy2* является полезным инструментом для морфологического анализа русского языка. Она предоставляет удобные функции для лемматизации, склонения и получения грамматических характеристик слов, что может быть полезно во множестве задач обработки естественного языка.

В нашей задаче реализации словарного фильтра с помощью *Rymorphy2* будем получать все возможные формы слов, находящихся в словаре запрещённых

					ПГУ1.090301.11.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		25

слов. Это позволит администратору задавать только начальные формы слова. Для того, чтобы администратор мог редактировать список запрещённых слов, определим обработчики команд *add_ban_word* и *remove_ban_word* для добавления и удаления запрещённых слов соответственно, которые будут вызываться при получении от администратора соответствующих команд. В этих обработчиках

опишем

отделение слова от команды и добавление или удаление соответственно слова из базы данных. Также опишем функцию *on_message*, отвечающую за обработку сообщений. Она будет вызывать обработку команд, если сообщение является командой, список которых задаётся в модуле *config.py*, и проверять сообщение на наличие запрещённых слов в остальных случаях. Для этого подключаемся к базе данных *PostgreSQL* при помощи библиотеки *psycopg2* и загружаем оттуда список

запрещённых слов. Получаем все возможные словоформы для каждого из загруженных слов и ищем их в сообщении. Если ни одного из запрещённых слов в любой форме не находится, сообщение отправляется в текстовый канал, а пользователю, являющемуся автором сообщения, увеличивается количество "очков опыта" на единицу. Это значение хранится в поле *score* записи пользователя в базе данных. Затем вызовем функцию проверки достижения уровня *check_lvl*, которая выдаст пользователю новый уровень, если у него стало достаточно "очков опыта". Если же запрещённое слово было найдено, пользователю выдаётся предупреждение, число которых хранится в поле *warnings* его записи в базе данных. Если число предупреждений достигло трёх, пользователю выдаётся временный запрет на использование каналов, и вызывается функция *write_ban_sleep*, отсчитывающая время до возвращения пользователю его возможностей. Такой же запрет может быть выдан администратором сервера при помощи команды *write_ban*, причём администратор может указать время, на которое пользователь будет заблокирован.

В этом же модуле реализована функция *on_voice_state_update*, которая сохраняет время захода пользователя в голосовые каналы в поле *voice_time* его записи в базе данных и начисляет очки опыта по количеству минут, проведённых

					ПГУ1.090301.11.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		26

в канале, после смены канала или выхода из него. После начисления "очков" вызывается функция *check_lvl*.

Также в модуле *main.py* определена функция *on_member_join*, которая обрабатывает подключение к серверу нового пользователя. Она сохраняет дату и время первого подключения пользователя к серверу, присваивает ему роль первого уровня и создаёт запись о нём в базе данных.

1.7.2 Разработка модуля *write_ban_thread.py*

В этом модуле реализуется функция *write_ban_sleep*, которая возвращает пользователю роль после истечения времени блокировки. Для отсчёта времени блокировки используется функция *asyncio.sleep*. Для возвращения пользователю его роли функция *write_ban_sleep* читает из его записи в базе данных поле *lvl*, в котором хранится уровень пользователя, после чего присваивает пользователю роль, соответствующую его уровню. Для того, чтобы функция *write_ban_sleep* не блокировала работу бота, используется библиотека *asyncio*, позволяющая функции работать асинхронно.

1.7.3 Разработка модуля *lvl_inspector.py*

В этом модуле реализована функция *check_lvl*, которая изменяет роль пользователя в соответствии с количеством "очков опыта", которое она читает из поля *score* записи пользователя в базе данных. В этой функции задаются значения "очков опыта", необходимые для перехода на определённый уровень. Помимо присвоения нужной роли, функция сохраняет уровень пользователя в поле *lvl* его записи в базе данных.

1.7.4 Разработка модуля *config.py*

В этот модуль вынесены токен для подключения бота, данные для подключения базы данных, список доступных команд и время автоматической блокировки.

					ПГУ1.090301.11.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		27

1.8 Тестирование и отладка

Убедимся в том, что все поставленные задачи выполнены, все необходимые функции работают корректно. Для этого развернём бота на сервере и подключим к серверу *Discord*. Создание сервера Discord изображено на рисунках 7, 8, 9, 10.

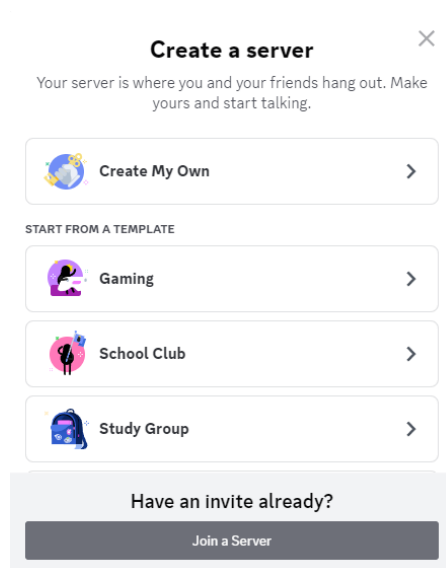


Рисунок 7 – Диалоговое окно создания сервера

Выбираем первый пункт и создаём сервер без использования шаблонов.

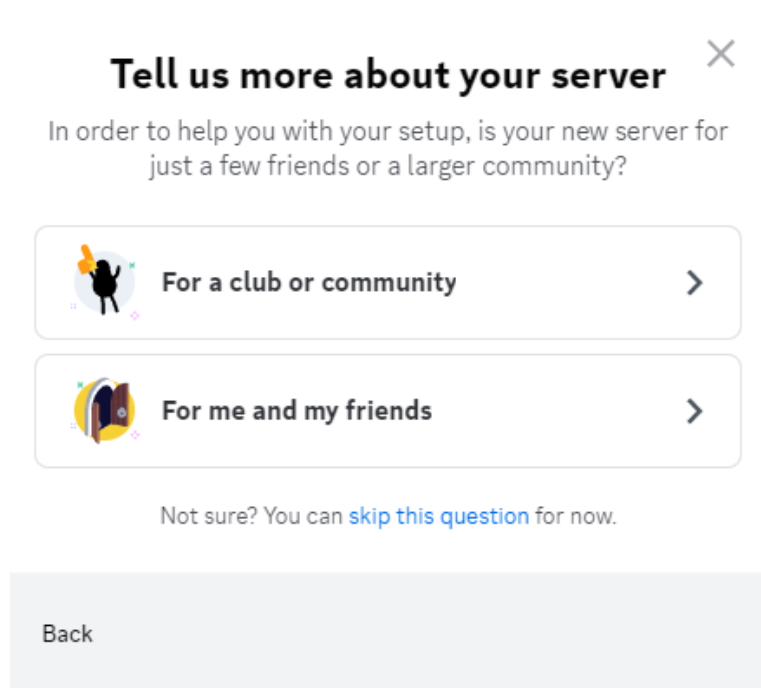


Рисунок 8 – Окно выбора типа сервера

					ПГУ1.090301.11.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		28

Выбираем тип сервера «для меня и друзей».

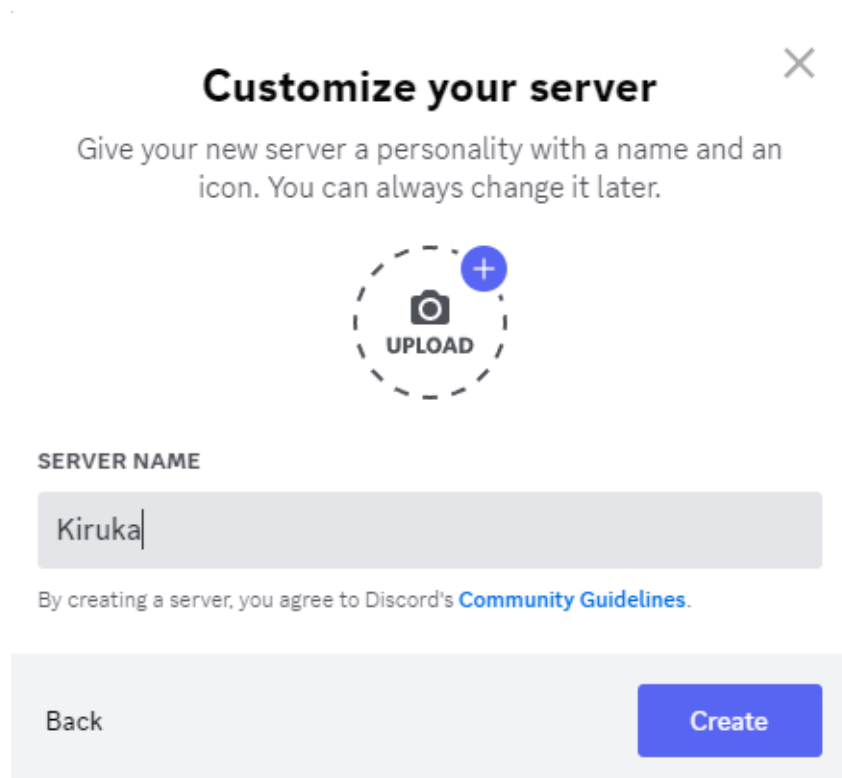


Рисунок 9 – Окно выбора имени сервера

Вводим имя сервера.

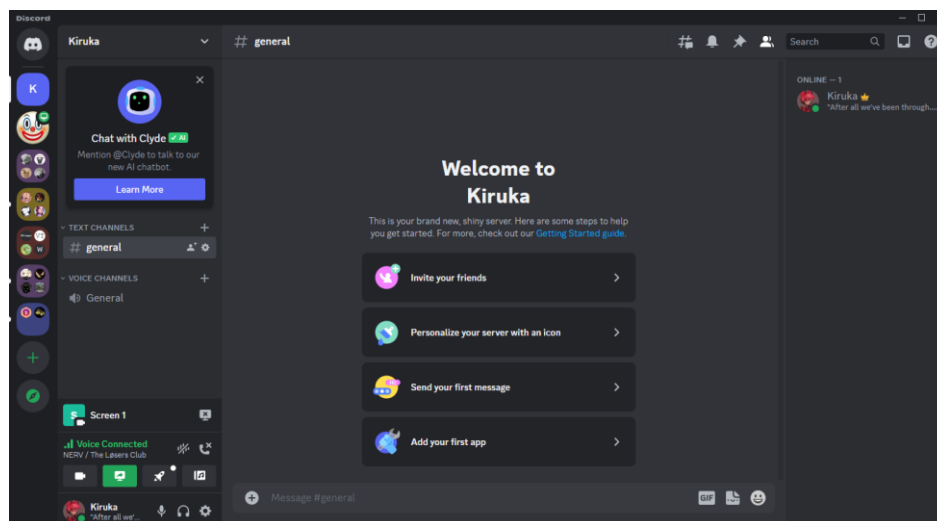


Рисунок 10 – Сервер создан

Сервер создан. По умолчанию в него уже добавлены два канала — текстовый и голосовой. Теперь получим идентификатор канала, для чего должен быть включён режим разработчика.

					ПГУ1.090301.11.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		29

Чтобы создать бота, перейдём на специальную страницу *Discord* для разработчиков: <https://discord.com/developers/>. Откроем её в соседней вкладке браузера, не выходя из учетной записи *Discord*.

Создадим приложение, затем в этом приложении создадим бота и настроим для него разрешения, а затем добавим бота на сервер. Создание приложения изображено на рисунке 11.

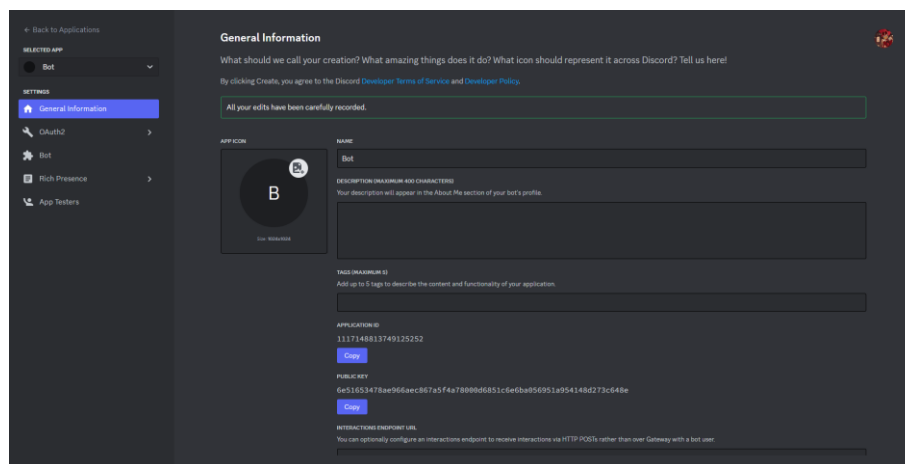


Рисунок 11 – Создание приложения

Создание бота изображено на рисунке 12.

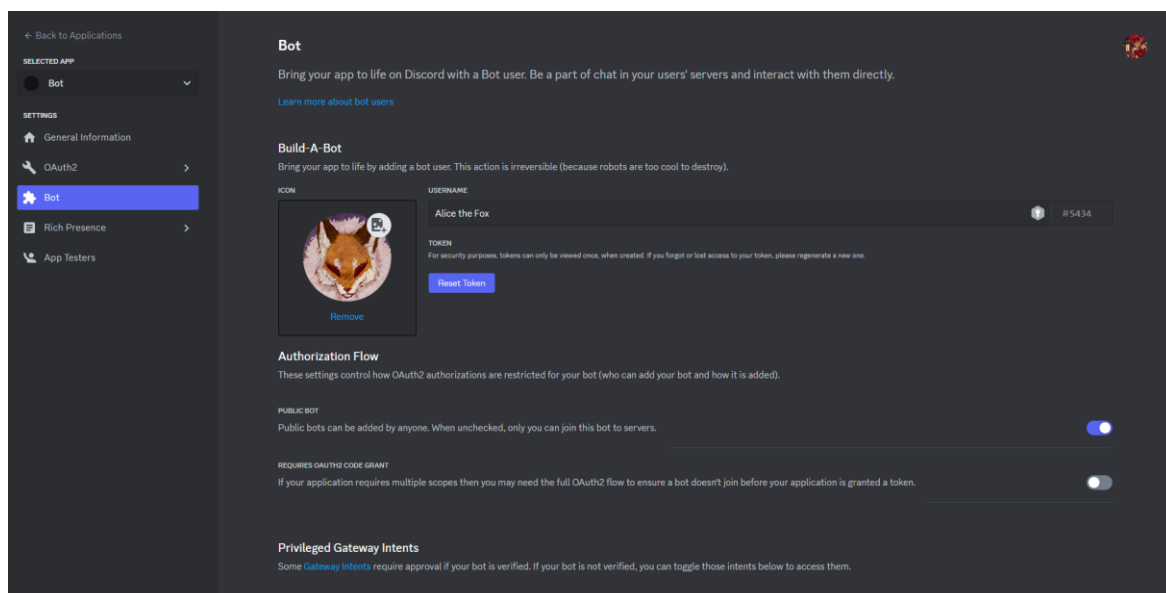


Рисунок 12 – Создание бота

На рисунке 13 изображены настройки авторизации бота.

					ПГУ1.090301.11.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		30

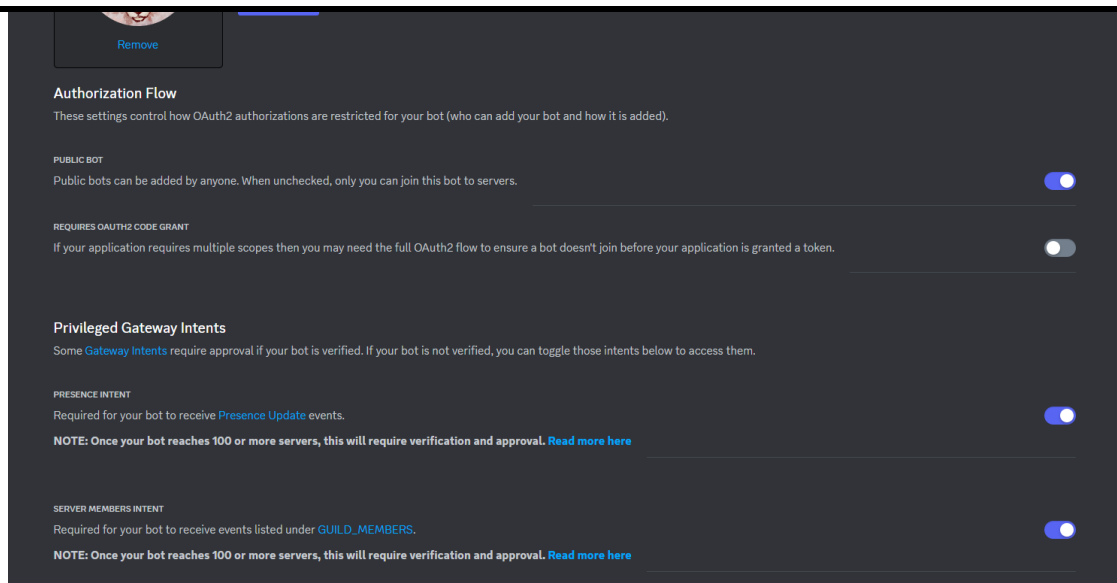


Рисунок 13 – Настройки авторизации бота

Поскольку наш бот будет выполнять роль помощника администратора, выдадим ему права администратора. Это продемонстрировано на рисунке 14.

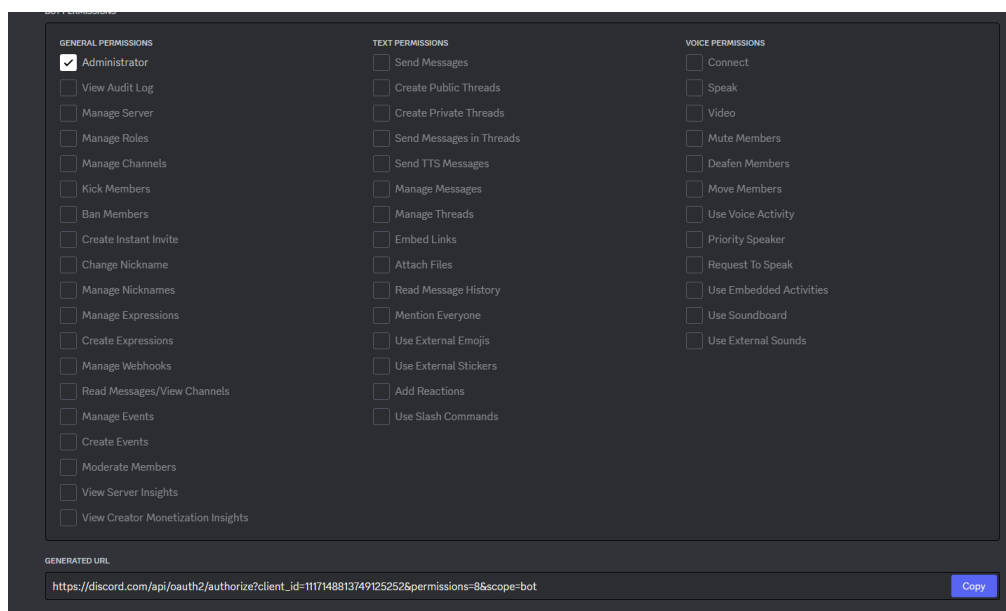


Рисунок 14 – Разрешения бота

					ПГУ1.090301.11.001 ПЗ	Лист
						31
Изм.	Лист	№ докум.	Подпись	Дата		

После выдачи боту прав *Discord* предоставляет ссылку, перейдя по которой, можно добавить бота на сервер. Это изображено на рисунке 15.

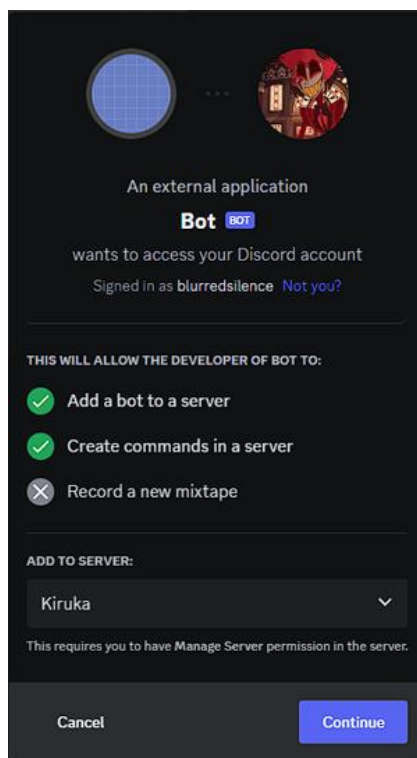


Рисунок 15 – Добавление бота на сервер

Подтверждаем разрешения бота, что изображено на рисунке 16.

					ПГУ1.090301.11.001 ПЗ	Лист
						32
Изм.	Лист	№ докум.	Подпись	Дата		

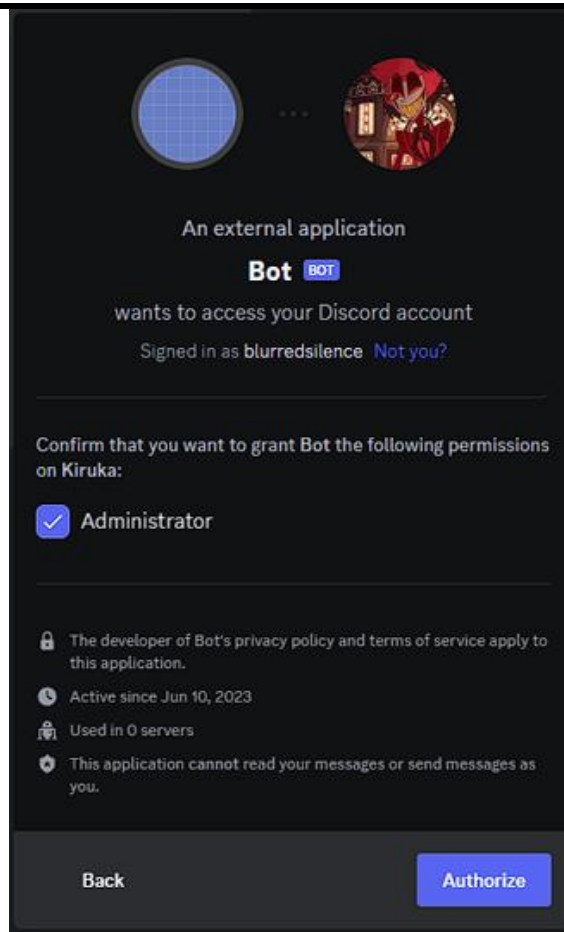


Рисунок 16 – Подтверждение разрешений бота

На рисунке 17 изображён сервер с добавленным на него ботом.

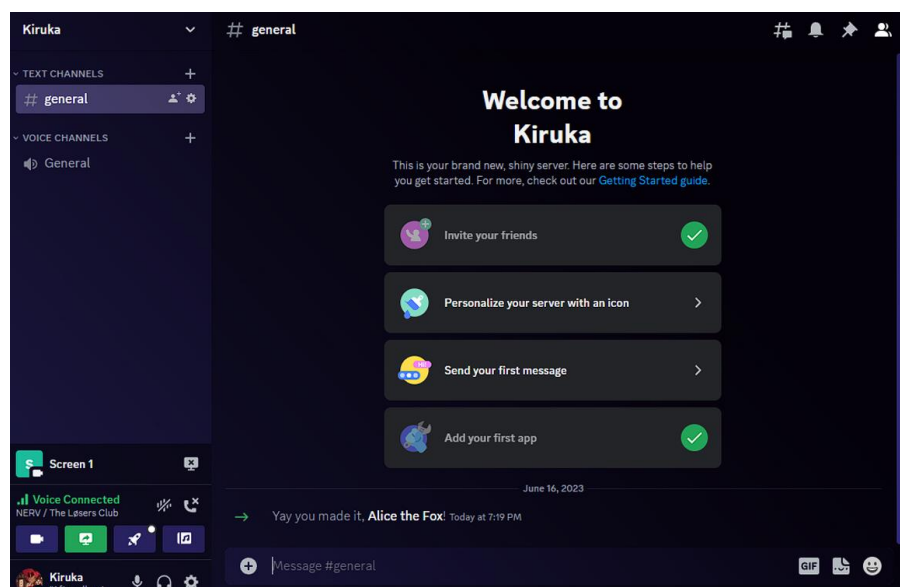


Рисунок 17 – Сервер с добавленным на него ботом

					ПГУ1.090301.11.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		33

Теперь бот добавлен на сервер и можно посмотреть его профиль, который приведён на рисунке 18.

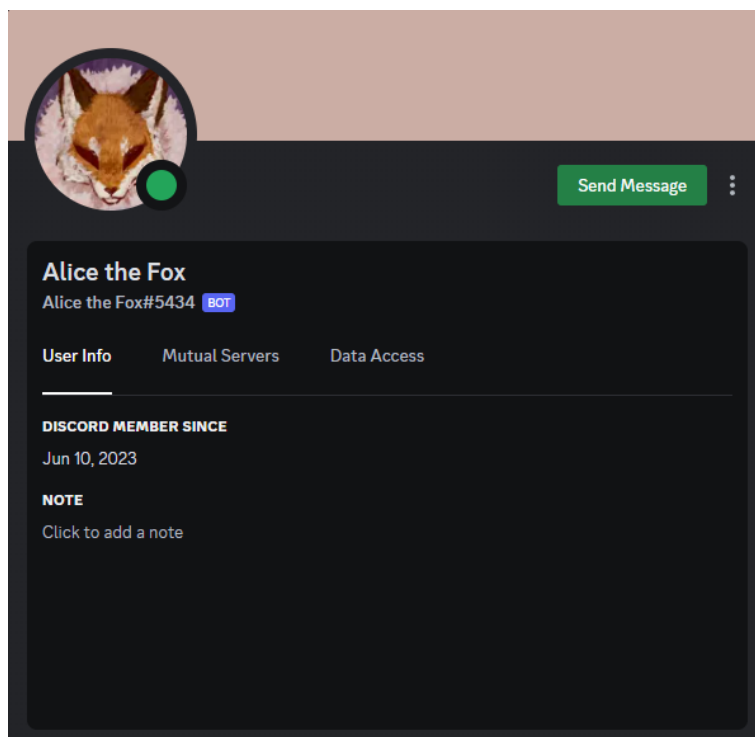


Рисунок 18 – Профиль бота

Для того, чтобы бот мог выдавать роли, нужно создать их на сервере. Роли изображены на рисунке 19.

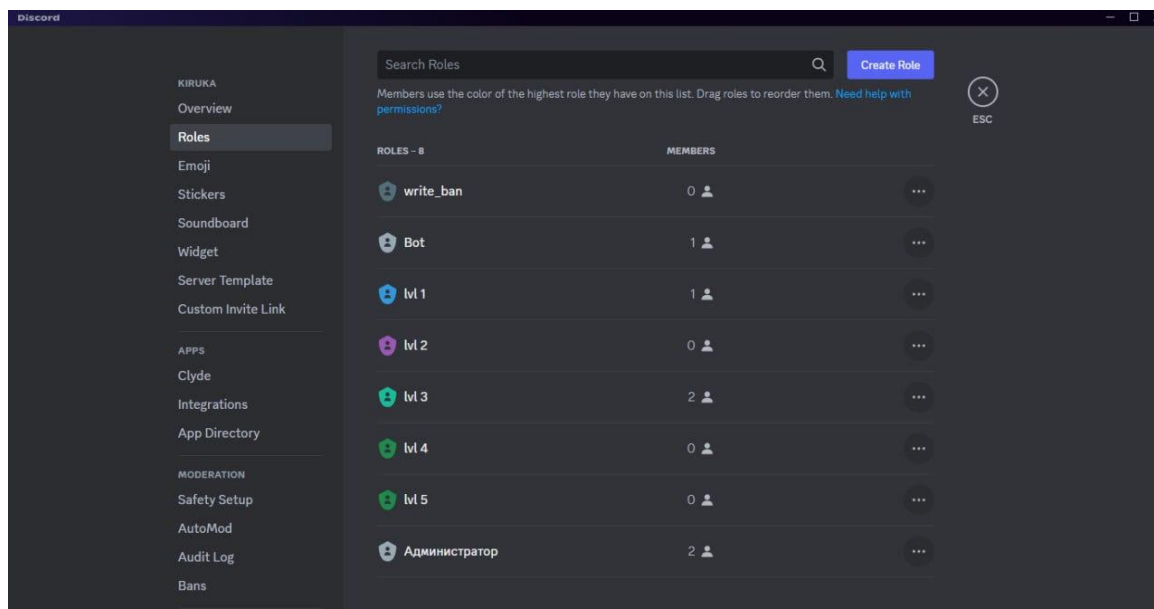


Рисунок 19 – Роли на сервере

Теперь можно тестировать функции бота.

На рисунке 20 изображено назначение ботом пользователю роли первого уровня при первом подключении на сервер. На рисунке 21 видно, что роль пользователю присвоена.

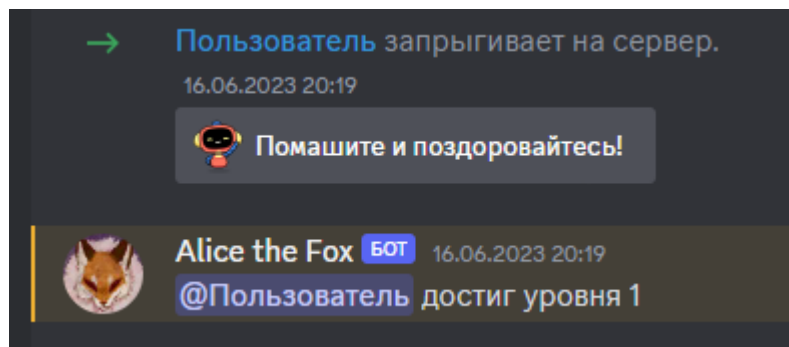


Рисунок 20 – Бот присваивает роль пользователю

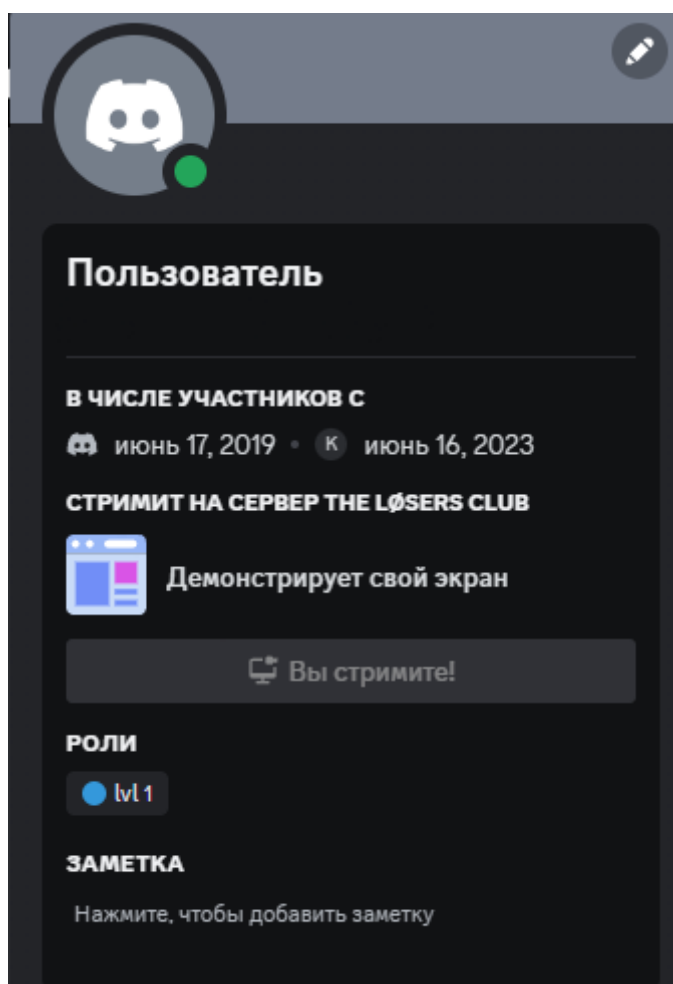


Рисунок 21 – Роль присвоена пользователю

Теперь администратор при помощи команды блокирует пользователя на 0,01 часа. Это изображено на рисунке 22. Роль первого уровня снимается с пользователя и ему выдаётся роль *write_ban*, запрещающая отправку сообщений и подключение к голосовым каналам.

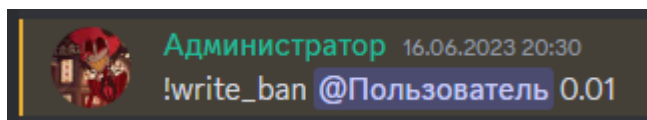


Рисунок 22 – Администратор блокирует пользователя

Выданную роль можно увидеть на рисунке 23.

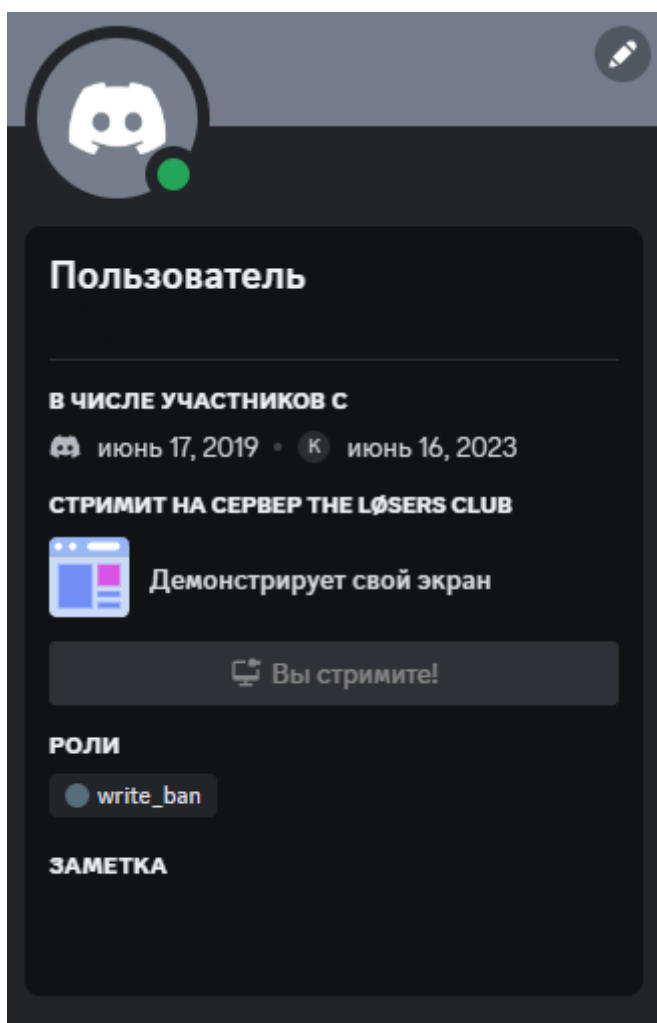


Рисунок 23 – Пользователь получил роль заблокированного

Спустя указанное время роль пользователя первого уровня восстановилась.

Теперь протестируем систему очков опыта. Для того, чтобы выяснить, какое количество очков опыта имеет пользователь, он должен ввести команду `!хр`. Работа этой команды изображена на рисунке 24.

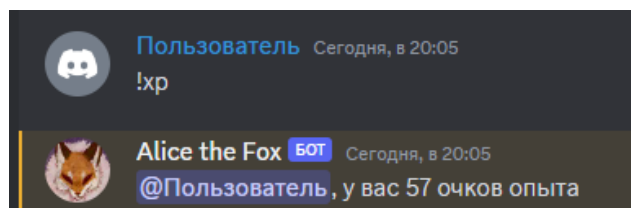


Рисунок 24 – Пользователь узнал свою статистику

При превышении порога опыта бот поднимает уровень участника сервера. На рисунке 25 изображено сообщение от бота об этом, а на рисунке 26 — отображение новой роли.

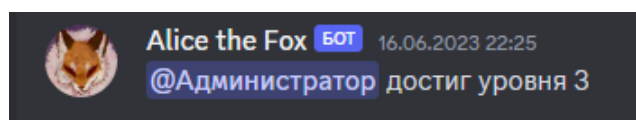


Рисунок 25 – Сообщение о повышении уровня

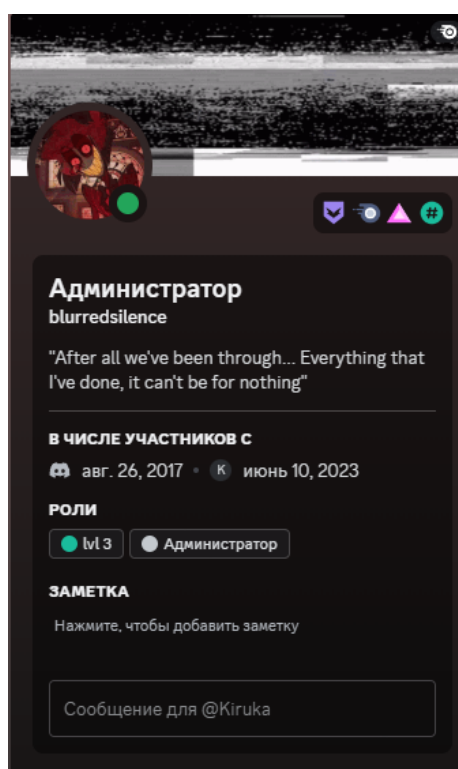


Рисунок 26 – Отображение уровня в профиле

Теперь проверим работу настраиваемого фильтра слов. На рисунке 27 изображено, как администратор добавляет слово в список запрещённых слов. Для примера возьмём слово «физрук».

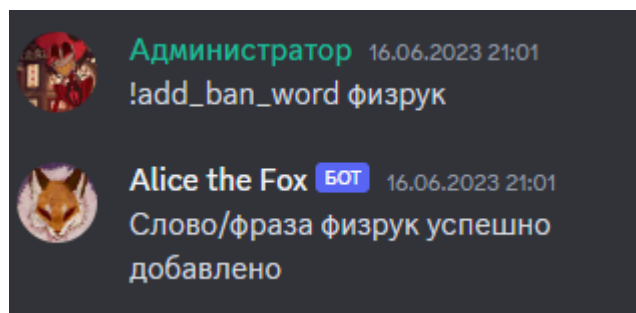


Рисунок 27 – Добавление слова в список запрещённых слов

Теперь пользователь получает предупреждение, если попытается написать это слово. Как видно на рисунке 28, отличия в форме слова не мешают боту распознать в нём запрещённое.

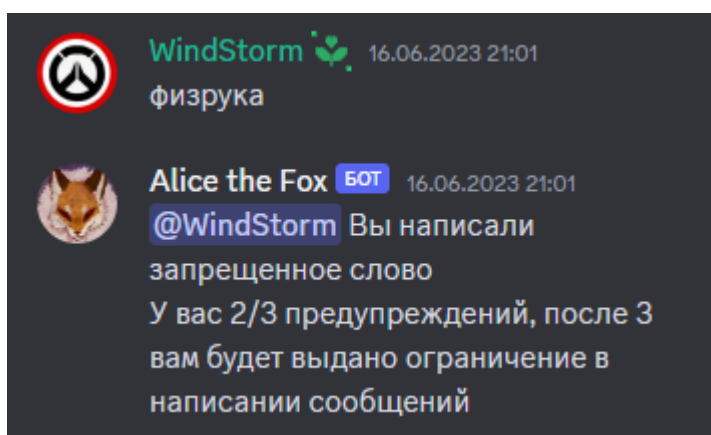


Рисунок 28 – Пользователь получает предупреждение

Таким образом, мы убедились, что все функции бота работают в соответствии с поставленными задачами.

2 Охрана труда

2.1 Анализ негативных факторов, воздействующих на оператора ПЭВМ и способы их устранения

Работа с вычислительной техникой непременно сопровождается ограниченной подвижностью и длительным статическим мышечным напряжением, а также может привести к серьезному перенапряжению зрительных органов. Эти неблагоприятные факторы приводят к переутомлению, снижению работоспособности, психологическому раздражению, появлению физических недомоганий и общей усталости.

Что бы снизить количество вредоносного воздействия средств вычислительной техники на здоровье человека необходимо придерживаться ряда правил использования ПЭВМ. Основными, из которых являются:

- правильная организация рабочего места;
- обеспечение достаточной освещенности рабочего места;
- соблюдение правил электробезопасности;
- соблюдение правил пожарной безопасности;
- регуляция режима труда и отдыха.

2.1.1 Оборудование рабочего места

Основные предъявляемые требования к рабочему месту:

- расстояние от глаз до экрана должно быть в пределах 600-700 мм;
- монитор необходимо расположить ниже уровня глаз под углом 10-15 градусов к направлению прямого взгляда;
- высота рабочего стола должна регулироваться в пределах 680-800 мм. В случае отсутствия такой возможности, высота стола должна составлять 725 мм;
- клавиатуру следует размещать на поверхности стола на расстоянии 100-300 мм от края, обращенного к пользователю, или на специальной регулируемой по высоте рабочей поверхности;

					ПГУ1.090301.11.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		39

- рабочий стол должен иметь достаточное пространство для ног, включая высоту не менее 600 мм, глубину на уровне колен не менее 450 мм и на уровне вытянутых ног не менее 650 мм.

На рисунке 29 изображена организация рабочего места, которая отвечает указанным требованиям.

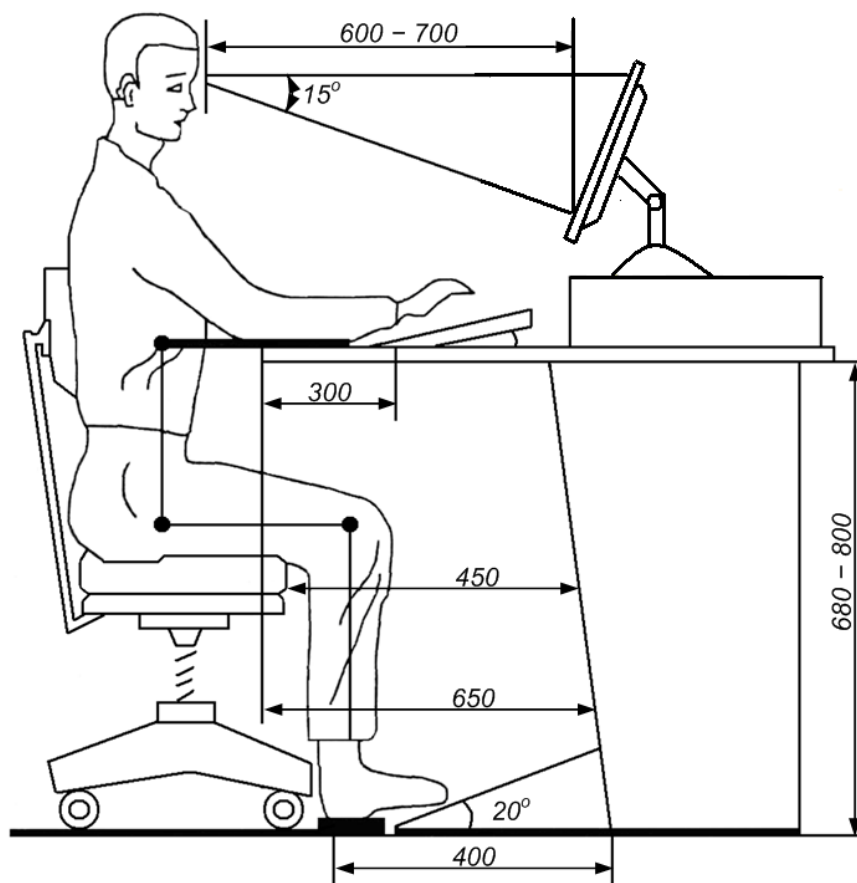


Рисунок 29 – Организация рабочего места

Важно, чтобы рабочее место с дисплеем обеспечивало удобство оператора в положении сидя и не создавало перегрузки костно-мышечной системы.

Организовав правильно рабочее место, также необходимо обеспечить правильную посадку пользователя ПК, правильная посадка изображена на рисунке 30.



Рисунок 30 – Правильная посадка за ПК

Правильная посадка пользователя ПК обеспечивает выполнение следующих условий:

- запястья должны быть в прямом положении;
- плечи должны быть расслаблены;
- локти должны полностью лежать на столе, перпендикулярно полу (угол в локтевых суставах составляет 90-100 градусов).

2.1.2 Освещенность рабочего места

Стандарты освещенности определяются, учитывая характер работы, место и поверхность. Для пультов ЭВМ и дисплеев рекомендуется комбинированное освещение не менее 400 люкс и общее освещение не менее 200 люкс.

На рабочем месте необходимо обеспечивать равномерную яркость, исключая яркие и блестящие предметы. Для разнообразия в поле зрения рекомендуется использовать отдельные яркие поверхности.

В основном для освещения используются потолочные или встроенные светильники с люминесцентными лампами. Чтобы избежать засветки экранов,

					ПГУ1.090301.11.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		41

светильники общего освещения следует устанавливать сбоку от рабочего места, параллельно линии зрения оператора.

2.1.3 Электробезопасность

Требования к электробезопасности регламентируются ГОСТ 12.1.019-2017 "ЭЛЕКТРОБЕЗОПАСНОСТЬ".

ПК работает от сети переменного тока 50 Гц с напряжением 230 В, что представляет опасность для жизни. При работе на ПК необходимо соблюдать следующие требования:

- все устройства должны быть подключены к одной фазе электрической сети;
- питание устройств ПК должно осуществляться только через розетки типа "Европа" с заземляющими контактами;
- все розетки для подключения устройств ПК должны иметь маркировку по напряжению;
- номинальное напряжение сети должно быть нанесено на стену или щит крупными символами яркой краской (высотой не менее 50 мм) возле розетки.

При использовании вычислительной техники и периферийного оборудования необходимо проявлять осторожность в отношении электропроводки и приборов, помнить, что нарушение правил безопасности может привести к угрозе здоровью и жизни человека.

Необходимо регулярно следить за исправностью электропроводки, выключателей и розеток, используемых для подключения оборудования.

При обнаружении неисправностей немедленно отключите электрооборудование. Продолжение работы возможно только после устранения всех неисправностей.

2.1.4 Пожарная безопасность

Наиболее распространенными причинами пожаров при использовании электроустановок и ПК являются:

- короткое замыкание проводов и электрических устройств;

					ПГУ1.090301.11.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		42

- воспламенение горючих материалов рядом с электрическими устройствами;

- большие переходные сопротивления в контактных соединениях.

Профилактические меры по пожарной безопасности проводятся в соответствии с федеральным законом от 21.12.1994 №69-ФЗ (ред. от 28.09.2010) "О пожарной безопасности". Для обеспечения пожарной безопасности необходимо принять ряд организационных мероприятий, направленных на обеспечение безопасности людей, предотвращение пожара и его распространения, а также создание условий для эффективного тушения пожаров. Для повышения противопожарной безопасности помещений следует принимать следующие меры:

- проходы и запасные выходы не должны быть загромождены предметами, препятствующими свободному прохождению;

- средства пожаротушения должны быть всегда доступны;

- система вентиляции должна быть оснащена устройствами, автоматически отключающими ее при пожаре, а также огнезадерживающими клапанами;

- автоматическая пожарная сигнализация и средства пожаротушения должны быть в исправном состоянии и соответствовать типовым правилам технического содержания установок пожарной автоматики.

2.1.5 Режим труда и отдыха

Работник, занятый работой с ПК, должен соблюдать режим труда и отдыха в зависимости от продолжительности, вида и категории трудовой деятельности. Различаются следующие виды трудовой деятельности, связанные с использованием персонального компьютера:

- группа А. Работа по чтению информации с экрана видеодисплейного терминала с предварительным запросом;

- группа Б. Работа по вводу информации;

- группа В. Творческая работа в режиме диалога с ПЭВМ.

					ПГУ1.090301.11.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		43

При выполнении работ, относящихся к различным видам трудовой деятельности, основной вид деятельности с ПК определяется как тот, который занимает не менее 50% времени в течение всей рабочей смены или рабочего дня.

Для каждого вида трудовой деятельности устанавливаются 3 категории тяжести и напряженности работы с ПК:

- для группы А категория определяется суммарным числом считываемых знаков за рабочую смену (не более 60000 знаков за смену);
- для группы Б категория определяется суммарным числом считываемых или вводимых знаков за рабочую смену (не более 40000 знаков за смену);
- для группы В категория определяется суммарным временем прямой работы с ПК за рабочую смену (не более 6 часов за смену).

Таблица 2.1 отображает общее время перерывов для каждой категории и вида работы.

Таблица 2.1 – Время перерывов

Категория работы с ПЭВМ или ВДТ	Уровень нагрузки за рабочую смену при видах работы с ВДТ			Суммарное время регламентированных перерывов, мин	
	группа А, количество знаков	группа Б, количество знаков	группа В час	при 8-часовой смене	при 12-часовой смене
1	До 20000	До 15000	До 2	50	80
2	До 40000	До 30000	До 4	70	110
3	До 60000	До 40000	До 6	90	140

Заключение

В ходе выполнения выпускной квалификационной работы был реализован бот для системы *Discord*, с использованием языка *Python*, как основного языка программирования. Был реализован фильтр слов с использованием библиотеки для морфологического анализа *py morphology2*.

Реализованный бот облегчает администрирование на сервере, помогает сохранять дружественную и безопасную атмосферу.

Одним из достоинств реализованного бота является наличие удобного фильтра слов и наличие выдачи системы уровней.

В дальнейшем можно доработать реализованного бота, расширив функциональность бота, например, дать возможность администратору добавлять и изменять при помощи команд роли и условия их получения. Также дать возможность пользователям проверять свою статистику: количество предупреждений и "очков опыта". Переработать систему блокировки таким образом, чтобы она возвращала роль пользователя после истечения времени блокировки, даже если бот был перезапущен.

					ПГУ1.090301.11.001 ПЗ	Лист
						45
Изм.	Лист	№ докум.	Подпись	Дата		

Список использованных источников

1. Бхаргава А. Грожаем алгоритмы. Иллюстрированное пособие для программистов и любопытствующих. — СПб.: Питер, 2017. — 288 с.: ил.
2. Моргунов Е.П. PostgreSQL. Основы языка SQL: учеб. Пособие / Е.П. Моргунов; под ред. Е.В. Рогова, П.В. Лузанова. – СПб.: БХВ-Петербург, 2018. – 336 с.: ил.
3. Дж. Рамбо, М. Блаха *UML 2.0*. Объектно-ориентированное моделирование и разработка, 2-е издание. – СПб.: Питер, 2007 — 544 с.
4. Руководство по стилю кода *Python*. [Электронный ресурс] / Режим доступа: <https://peps.python.org/pep-0008/> свободный. (дата обращения: 10.04.2023).
5. Документация к *Discord API*. [Электронный ресурс] / Режим доступа: <https://discord.com/developers/docs/reference> свободный. (дата обращения: 20.04.2022).

					ПГУ1.090301.11.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		46

Приложение А

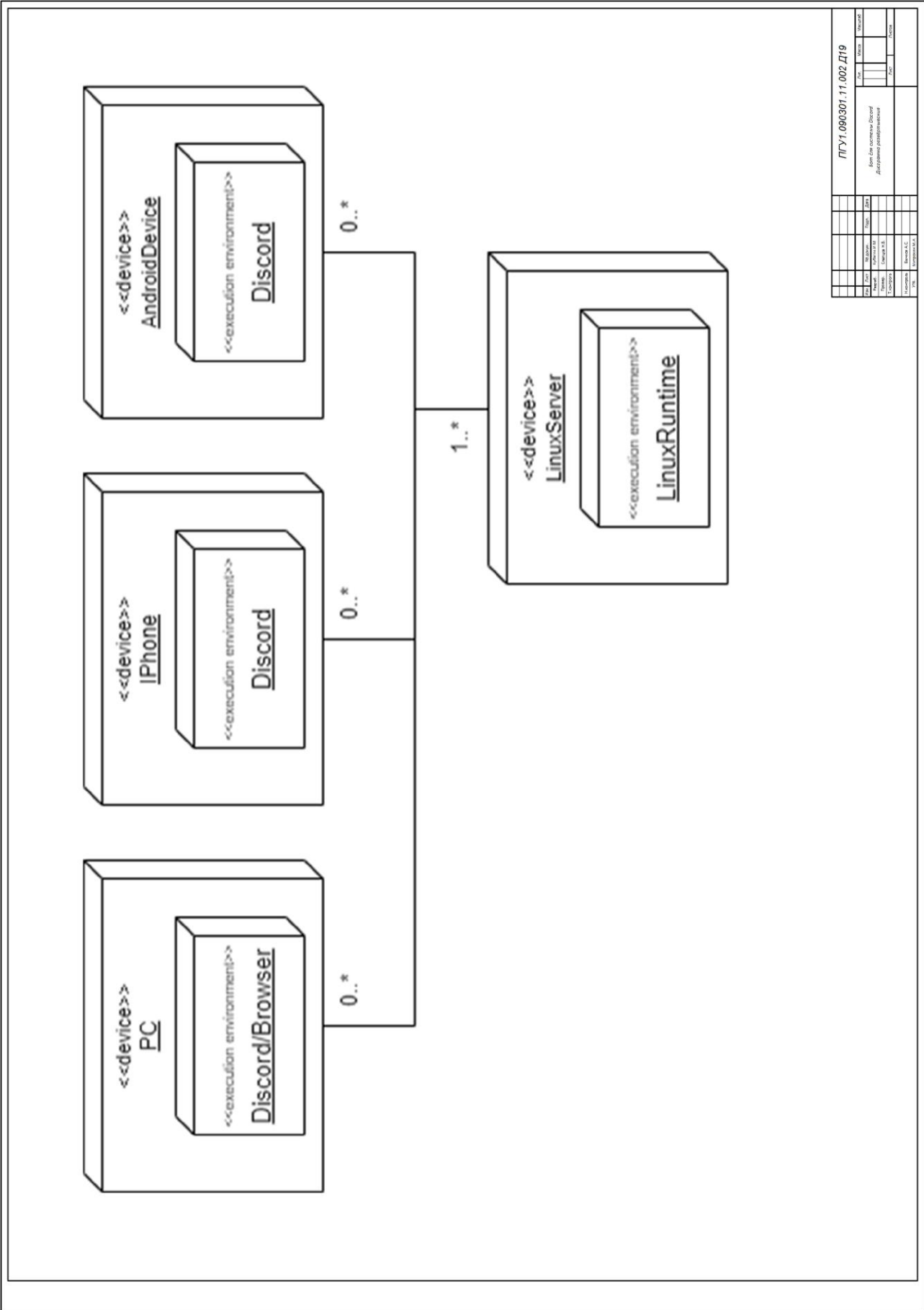
(обязательное)

Бот для системы *Discord*

UML диаграммы

					ПГУ1.090301.11.001 ПЗ	Лист
						47
Изм.	Лист	№ докум.	Подпись	Дата		

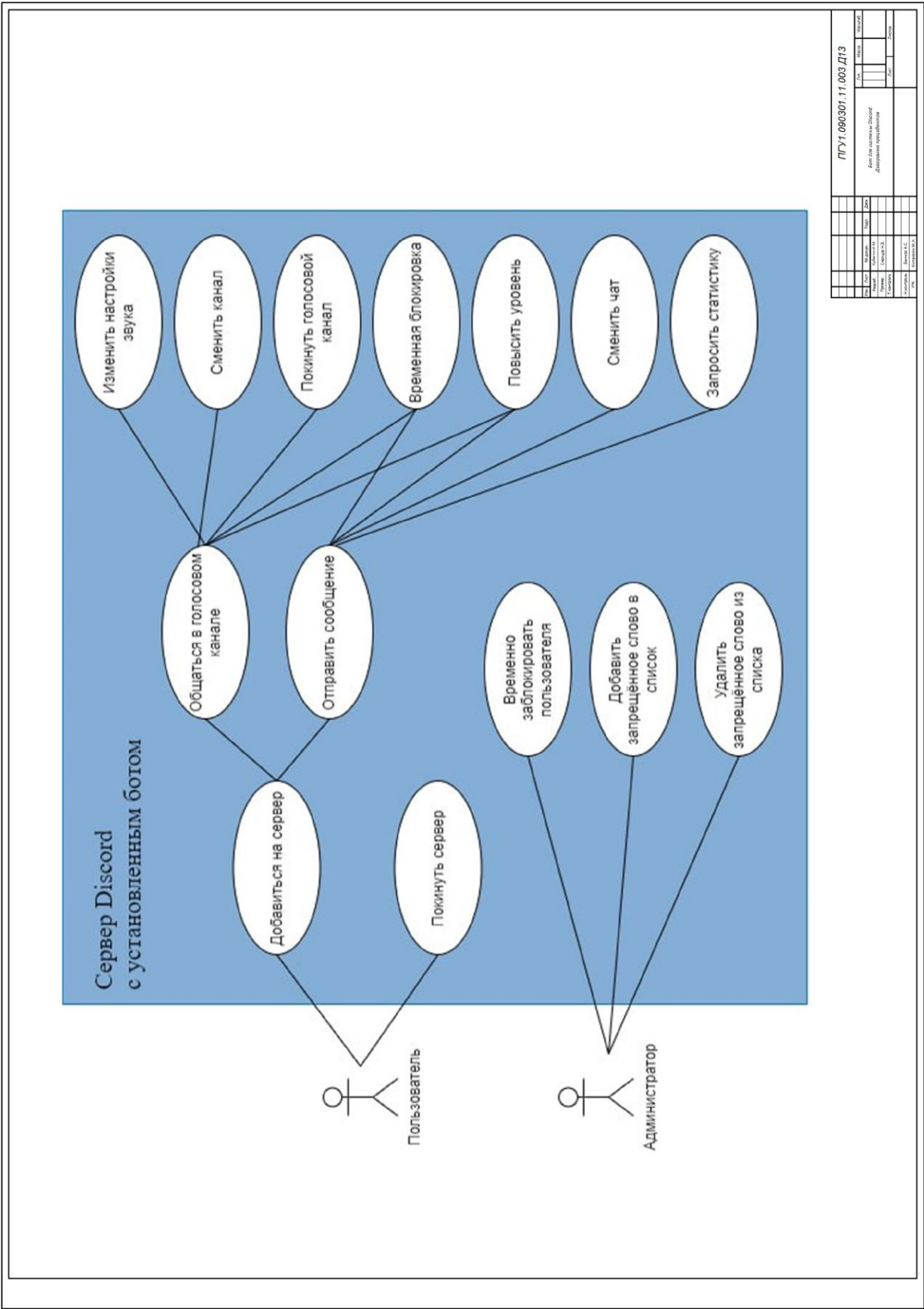
Приложение А. 1 – UML – диаграмма развертывания



ПГУ1.090301.11.002 Д19									
№	С	В	М	Д	Г	И	О	Д	С
Вопрос системы Discord									
Диагностика развертывания									
Имя	Имя	Имя	Имя	Имя	Имя	Имя	Имя	Имя	Имя
Фамилия	Фамилия	Фамилия	Фамилия	Фамилия	Фамилия	Фамилия	Фамилия	Фамилия	Фамилия
Пол	Пол	Пол	Пол	Пол	Пол	Пол	Пол	Пол	Пол
Дата	Дата	Дата	Дата	Дата	Дата	Дата	Дата	Дата	Дата

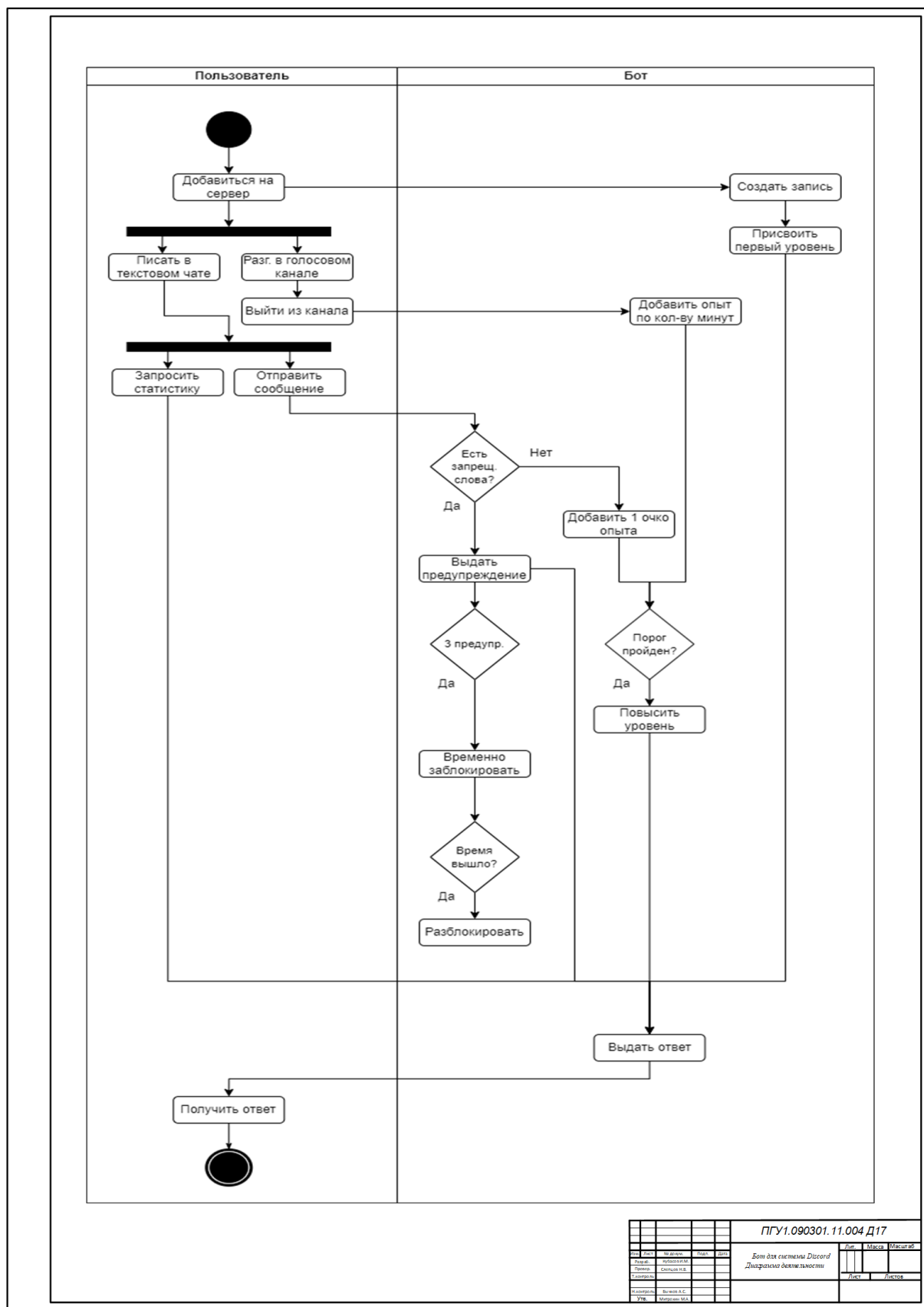
Изм.	Лист	№ докум.	Подпись	Дата

Приложение А.2 – UML – диаграмма прецедентов

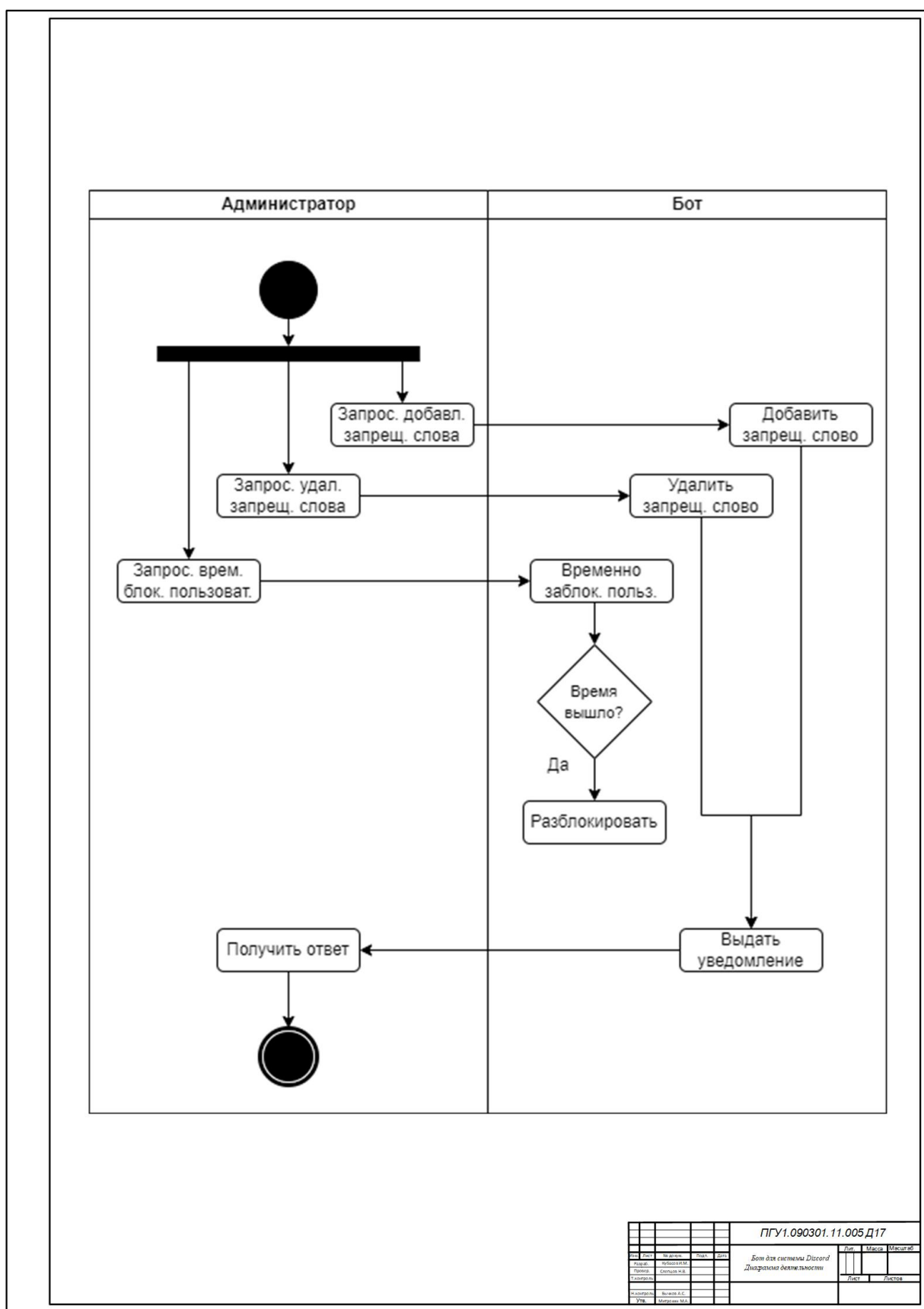


ПГУ1.090301.11.003 ДПЗ									
Вопросник по оценке качества разработки программного обеспечения									
№	Вопрос	Да	Нет	Не знаю	Не applicable	Не applicable	Не applicable	Не applicable	Не applicable
1	Вопрос 1								
2	Вопрос 2								
3	Вопрос 3								
4	Вопрос 4								
5	Вопрос 5								
6	Вопрос 6								
7	Вопрос 7								
8	Вопрос 8								
9	Вопрос 9								
10	Вопрос 10								

Приложение А.3 –UML – диаграмма деятельности для пользователя



Приложение А.4 –UML – диаграмма деятельности для администратора



ПГУ1.090301.11.005 Д17									
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия
Подпись	Подпись	Подпись	Подпись	Подпись	Подпись	Подпись	Подпись	Подпись	Подпись
Инициалы	Инициалы	Инициалы	Инициалы	Инициалы	Инициалы	Инициалы	Инициалы	Инициалы	Инициалы
Инициалы	Инициалы	Инициалы	Инициалы	Инициалы	Инициалы	Инициалы	Инициалы	Инициалы	Инициалы

Бот для системы Discord
Диаграмма деятельности

Лист 1
Листа 1

Изм.	Лист	№ докум.	Подпись	Дата
------	------	----------	---------	------

ПГУ1.090301.11.001 ПЗ

Лист

51

Приложение Б

(обязательное)

Бот для системы *Discord*

Листинг программы

					ПГУ1.090301.11.001 ПЗ	Лист
						52
Изм.	Лист	№ докум.	Подпись	Дата		

Файл *Main.py*

```
import discord
from discord.ext import commands
import datetime
from discord.utils import get
import psycpg2
import pymorphy2

from config import *

from lvl_inspector import check_lvl
from write_ban_thread import write_ban_sleep

morph = pymorphy2.MorphAnalyzer()

# Подключаемся в БД
conn = psycpg2.connect(database=data_base, user=user,
password=password, host=host, port=port)

with conn.cursor() as cur:
    # Создаем таблицу для запрещенных слов
    try:
        cur.execute("""CREATE TABLE ban_words(
            id serial PRIMARY KEY,
            word varchar,
            server_id varchar)
        """)
        conn.commit()
    except:
        pass

    # Создаем таблицу для пользователей
    try:
        cur.execute("""CREATE TABLE users(
            id serial PRIMARY KEY,
            member_id varchar,
            server_id varchar,
            start_date varchar,
            score bigint,
            lvl int,
            voice_time varchar,
            warnings integer)
        """)
        conn.commit()
    except:
        pass

conn.close()

# создаем экземпляр бота
bot =
commands.Bot(command_prefix='!', intents=discord.Intents.all())
```

					ПГУ1.090301.11.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		53

```

@bot.command()
async def xp(ctx):
    conn = psycopg2.connect(database=data_base, user=user,
password=password, host=host, port=port)

    with conn.cursor() as cur:
        cur.execute("""SELECT score FROM users WHERE
member_id=%s AND server_id=%s""", (str(ctx.message.author.id),
str(ctx.guild.id)))
        score = cur.fetchone()
        score = score[0]
        await ctx.send(f'{ctx.message.author.mention}, у вас
{score} очков опыта')
    conn.close()

# Создаем команду для удаления запрещенного слова на сервер
@bot.command()
async def remove_ban_word(ctx):

    all_user_roles = []
    for role in ctx.message.author.roles:
        all_user_roles.append(role.name)

    if 'Администратор' not in all_user_roles:
        return False

    # Отделяем слово от общего текста сообщения
    word =
ctx.message.content.split('!remove_ban_word')[1].strip()

    # Если не указали слово то уведомляем в чат
    if word == '':
        await ctx.send('Вы не ввели слово/фразу')
    # Если все нормальн, удаляем слово из БД
    else:
        conn = psycopg2.connect(database=data_base, user=user,
password=password, host=host, port=port)

        with conn.cursor() as cur:
            cur.execute("""DELETE FROM ban_words WHERE word=%s
AND server_id=%s""", (word, str(ctx.guild.id)))
            conn.commit()
            await ctx.send(f'Слово/фраза {word} успешно
удалена')
        conn.close()

# Создаем команду для добавления запрещенного слова на сервер
@bot.command()
async def add_ban_word(ctx):

    all_user_roles = []
    for role in ctx.message.author.roles:
        all_user_roles.append(role.name)

```

					ПГУ1.090301.11.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		54

```

        if 'Администратор' not in all_user_roles:
            return False
        # Отделяем слово от общего текста сообщения
        word =
ctx.message.content.split('!add_ban_word')[1].strip()

        # Если не указали слово то уведомляем в чат
        if word == '':
            await ctx.send('Вы не ввели слово/фразу')
        # Если все нормальн, записываем слово в БД
        else:
            conn = psycopg2.connect(database=data_base, user=user,
password=password, host=host, port=port)

            with conn.cursor() as cur:
                cur.execute("""INSERT INTO ban_words (word,
server_id) VALUES (%s, %s)""", (word, ctx.guild.id))
                conn.commit()
                await ctx.send(f'Слово/фраза {word} успешно
добавлено')
            conn.close()

        # Создаем команду для запрета пользователю на время писать в
чате
        @bot.command()
        async def write_ban(ctx, member: discord.Member = None,
time:float = None):
            all_user_roles = []
            for role in ctx.message.author.roles:
                all_user_roles.append(role.name)

            if 'Администратор' not in all_user_roles:
                return False

            try:
                # Пробуем запустить функцию
                await write_ban_sleep(member, time)
            except Exception as e:
                print(e)
                # Если вернуло ошибку, уведомляем в чат
                await ctx.send('Вы неверно указали пользователя для
бана или время')

        # Проверяем что бот запустился
        @bot.event
        async def on_ready():
            print(f'{bot.user} has connected to Discord!')

        # Обрабатываем все входящие на сервер с ботом сообщения
        @bot.event
        async def on_message(message):

            # Получаем текст сообщения

```

					ПГУ1.090301.11.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		55

```

text = message.content

# Если сообщение от бота, то выходим
if message.author.bot == True:
    return

# конвертируем тип данных
member_id = str(message.author.id)
guild_id = str(message.author.guild.id)

# Проверка: написанный текст команда или нет
for comand in comand_list:
    # Если команда запускаем ее функцию и выходим
    if comand in text:
        await bot.process_commands(message)
        return

# Подключаемся к БД
conn = psycopg2.connect(database=data_base, user=user,
password=password, host=host, port=port)
with conn.cursor() as cur:
    # Получаем все запрещенные слова сервера
    cur.execute("""SELECT word FROM ban_words WHERE
server_id=%s""", (guild_id,))
    ban_words= cur.fetchall()

# Для каждого слова получаем список лексем
for word in ban_words:
    word = word[0]
    lexemes = morph.parse(word)[0].lexeme

# Если в тексте сообщения есть запрещенная лексема
for lexem in lexemes:
    if lexem.word in text:
        # Получаем какое по счету предупреждение
        cur.execute("""SELECT warnings FROM users
WHERE member_id=%s AND server_id=%s""", (member_id, guild_id))
        warnings = cur.fetchone()
        warnings = warnings[0]+1

# Если уже 3
if warnings == 3:
    # Уведомляем пользователя о нарушении
    правил
    await
message.author.guild.system_channel.send(
    f"{message.author.mention} Вы
написали запрещенное слово\n"
    f"У вас {warnings}/3
предупреждений\n"
    f"С этого момента вы не сможете
писать сообщения на протяжении {ban_time} часов")

# Сбрасываем кол-во предупреждений до 0
cur.execute("""UPDATE users SET
warnings=0 WHERE member_id=%s AND server_id=%s""",

```

					ПГУ1.090301.11.001 ПЗ	Лист
						56
Изм.	Лист	№ докум.	Подпись	Дата		

```

                                (member_id, guild_id))
                                conn.commit()

                                # Получаем текущий лвл пользователя
                                cur.execute("""SELECT lvl FROM users
WHERE member_id=%s AND server_id=%s""", (member_id, guild_id))
                                lvl = cur.fetchone()
                                lvl = lvl[0]

                                # Получаем роль и забираем ее у
пользователя
                                role = get(message.author.guild.roles,
name=f"lvl {lvl}")
                                await message.author.remove_roles(role)

                                # Получаем и присваиваем роль которая
запрещает писать сообщение
                                role = get(message.author.guild.roles,
name="write_ban")
                                await message.author.add_roles(role)

                                # Запускаем функцию которая через время
вернет пользователю возможность писать
                                await
write_ban_sleep(message.author, ban_time)

                                # Еще меньше 3
                                else:
                                    # Предупреждаем пользователя что он
нарушил
                                    await
message.author.guild.system_channel.send(f"{message.author.mention}
Вы написали запрещенное слово\n"

f"У вас {warnings}/3 предупреждений, после 3 вам будет выдано
ограничение в написании сообщений")
                                    cur.execute("""UPDATE users SET
warnings=warnings+1 WHERE member_id=%s AND server_id=%s""",
                                                (member_id, guild_id))

                                    conn.commit()
                                    return

                                # Если все нормально, то добавляем 1 очко лвл-а
пользователю
                                cur.execute("""UPDATE users SET score=score+1 WHERE
member_id=%s AND server_id=%s""", (member_id, guild_id))
                                conn.commit()

                                # Запускаем функцию проверки лвл-в
                                await check_lvl(message.author)
                                conn.close()

                                # Обрабатываем информацию об изменениях активности в голосовых
каналах

```

					ПГУ1.090301.11.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		57

```

@bot.event
async def on_voice_state_update(member, before, after):

    # Получаем данные о том с кем и где произошли изменения
    member_id = str(member.id)
    guild_id = str(member.guild.id)
    channel = after.channel

    # Подключаемся к БД
    conn = psycopg2.connect(database=data_base, user=user,
password=password, host=host, port=port)

    with conn.cursor() as cur:

        # Получаем время последнего присоединения в голосовой
канал
        cur.execute("""SELECT voice_time FROM users WHERE
member_id=%s AND server_id=%s""", (member_id, guild_id))
        voice_time = cur.fetchone()
        voice_time = voice_time[0]

        # Если пользователь перешел в канал а не вышел из него
if channel != None:
            # Если пользователь зашел в канал
            if voice_time is None:
                # Фиксируем и записываем в БД время подключения к
каналу
                time =
str(datetime.datetime.now()).split('.')[0]
                cur.execute("""UPDATE users SET voice_time=%s
WHERE member_id=%s AND server_id=%s""", (time, member_id, guild_id))

            # Если пользователь перешел из одного канала в
другой
            else:
                # Фиксируем время перехода
                time = datetime.datetime.now()

                # Считаем сколько времени прошло с момента
подключения в прошлый канал
                delta = time -
datetime.datetime.strptime(voice_time, '%Y-%m-%d %H:%M:%S')

                # Конвертируем время на очки лвл-а
                scores = int(delta.total_seconds() / 60)

                # Сохраняем очки в БД
                cur.execute("""UPDATE users SET score=score+%s
WHERE member_id=%s AND server_id=%s""",
                    (scores, member_id, guild_id))

                # Запускаем функцию проверки лвл
                await check_lvl(member)

            # Фиксируем и записываем в БД время подключения к
текущему каналу

```

					ПГУ1.090301.11.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		58

```

        time =
str(datetime.datetime.now()).split('.')[0]
        cur.execute("""UPDATE users SET voice_time=%s
WHERE member_id=%s AND server_id=%s""",
                    (time, member_id, guild_id))

    # Если пользователь вышел из канала
    else:
        # Фиксируем время перехода
        time = datetime.datetime.now()

        # Считаем сколько времени прошло с момента
подключения в прошлый канал
        delta = time -
datetime.datetime.strptime(voice_time, '%Y-%m-%d %H:%M:%S')

        # Конвертируем время на очки лвл-а
        scores = int(delta.total_seconds() / 60)

        # Сохраняем очки в БД
        cur.execute("""UPDATE users SET score=score+%s
WHERE member_id=%s AND server_id=%s""",
                    (scores, member_id, guild_id))

        # Запускаем функцию проверки лвл
        await check_lvl(member)

        # Указываем в БД время подключения NULL
        cur.execute("""UPDATE users SET voice_time=NULL
WHERE member_id=%s AND server_id=%s""",
                    (member_id, guild_id))

        # Сохраняем изменения в БД
        conn.commit()
    conn.close()

# Обрабатываем событие присоединения нового пользователя
@bot.event
async def on_member_join(member):

    # Фиксируем время когда присоединился пользователь
    time = str(datetime.datetime.now())
    cd = time.split('.')[0].replace(' ', '-').replace('-', ' ')
    cd = cd.replace(':', ' ')
    cd = cd.split(' ')
    start_date = f'{cd[2]}.{cd[1]}.{cd[0]} {cd[3]}:{cd[4]}'

    # Подключаемся к БД
    conn = psycopg2.connect(database=data_base, user=user,
password=password, host=host, port=port)

    with conn.cursor() as cur:

        # Получаем роль сервера и присваиваем ее

```

					ПГУ1.090301.11.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		59

```

        role = get(member.guild.roles, name="lvl 1")
        await member.add_roles(role)

        # Присылаем уведомление о достижении уровня
        await
member.guild.system_channel.send(f"{member.mention} достиг уровня
1")

        # меняем тип данных на строку
        member_id = str(member.id)
        guild_id = str(member.guild.id)

        # Записываем данные о пользователе в БД
        cur.execute("""INSERT INTO users (member_id, server_id,
start_date, score, lvl, warnings) VALUES
(%s,%s,%s,0,1,0)""", (member_id, guild_id, start_date))

        # Сохраняем изменения в БД
        conn.commit()
        conn.close()

bot.run(TOKEN)

```

Файл *Lvl_inspector.py*

```

from discord.utils import get
import psycopg2

from config import *

async def check_lvl(member):

    # Конвертируем тип данных
    member_id = str(member.id)
    guild_id = str(member.guild.id)

    # Подключаемся к БД
    conn = psycopg2.connect(database=data_base, user=user,
password=password, host=host, port=port)

    with conn.cursor() as cur:

        # Получаем текущие кол-во очков пользователя
        cur.execute("""SELECT score FROM users WHERE
member_id=%s AND server_id=%s""", (member_id, guild_id))
        score = cur.fetchone()
        score = int(score[0])

        # Получаем текущий лвл пользователя
        cur.execute("""SELECT lvl FROM users WHERE member_id=%s
AND server_id=%s""", (member_id, guild_id))
        lvl = cur.fetchone()

```

					ПГУ1.090301.11.001 ПЗ	Лист
						60
Изм.	Лист	№ докум.	Подпись	Дата		

```

        lvl = lvl[0]

        # Если выполняются условия то присваиваем новый
        уровень, а старый забираем
        if score >= 1000:
            if lvl != 5:
                role = get(member.guild.roles, name="lvl 5")
                await member.add_roles(role)

                role = get(member.guild.roles, name="lvl 4")
                await member.remove_roles(role)

                cur.execute("""UPDATE users SET lvl=5 WHERE
member_id=%s AND server_id=%s""", (member_id, guild_id))
                conn.commit()

                await
member.guild.system_channel.send(f"{member.mention} достиг уровня
5")

            elif score >= 250:
                if lvl != 4:
                    role = get(member.guild.roles, name="lvl 4")
                    await member.add_roles(role)

                    role = get(member.guild.roles, name="lvl 3")
                    await member.remove_roles(role)

                    cur.execute("""UPDATE users SET lvl=4 WHERE
member_id=%s AND server_id=%s""", (member_id, guild_id))
                    conn.commit()

                    await
member.guild.system_channel.send(f"{member.mention} достиг уровня
4")

                elif score >= 100:
                    if lvl != 3:
                        role = get(member.guild.roles, name="lvl 3")
                        await member.add_roles(role)

                        role = get(member.guild.roles, name="lvl 2")
                        await member.remove_roles(role)

                        cur.execute("""UPDATE users SET lvl=3 WHERE
member_id=%s AND server_id=%s""", (member_id, guild_id))
                        conn.commit()

                        await
member.guild.system_channel.send(f"{member.mention} достиг уровня
3")

                    elif score >= 50:
                        if lvl != 2:
                            role = get(member.guild.roles, name="lvl 2")
                            await member.add_roles(role)

                            role = get(member.guild.roles, name="lvl 1")
                            await member.remove_roles(role)

```

					ПГУ1.090301.11.001 ПЗ	Лист
						61
Изм.	Лист	№ докум.	Подпись	Дата		

```

        cur.execute("""UPDATE users SET lvl=2 WHERE
member_id=%s AND server_id=%s""", (member_id, guild_id))
        conn.commit()

        await
member.guild.system_channel.send(f"{member.mention} достиг уровня
2")

        conn.close()

```

Файл *Write_ban_thread.py*

```

import asyncio
from discord.utils import get
import discord
import psycopg2

from config import *

async def write_ban_sleep(member: discord.Member, time_ban:
float):

    # Подключаемся к БД
    conn = psycopg2.connect(database=data_base, user=user,
password=password, host=host, port=port)

    with conn.cursor() as cur:
        # Получаем уровень который был у пользователя
        cur.execute("""SELECT lvl FROM users WHERE member_id=%s
AND server_id=%s""", (str(member.id), str(member.guild.id)))
        lvl = cur.fetchone()
        lvl = lvl[0]

        # Возвращаем роль запрета
        role = get(member.guild.roles, name="write_ban")
        await member.add_roles(role)

        # Удаляем роль лвл-а
        role = get(member.guild.roles, name=f"lvl {lvl}")
        await member.remove_roles(role)

        # Ждем пока пройдет время запрета
        await asyncio.sleep(60 * 60 * time_ban)

    try:
        # Забираем роль запрета
        role = get(member.guild.roles, name="write_ban")
        await member.remove_roles(role)
    except:
        pass

    # Возвращаем роль лвл-а
    role = get(member.guild.roles, name=f"lvl {lvl}")

```

					ПГУ1.090301.11.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		62

```
await member.add_roles(role)
```

```
conn.close()
```

Файл *Config.py*

```
# Токен дискорд бота
```

```
TOKEN = ''
```

```
# данные подключения к БД
```

```
data_base = 'Kiruka_discord_bot'
```

```
user= 'windstorm'
```

```
password= '41206'
```

```
host= '104.219.250.25'
```

```
port= '5432'
```

```
# Список команд
```

```
comand_list = ('!write_ban',
```

```
'!add_ban_word', '!remove_ban_word', '!xp')
```

```
# Время запрета написания по умолчанию(за 3 нарушения)
```

```
ban_time = 2
```

					ПГУ1.090301.11.001 ПЗ	Лист
						63
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение В

(обязательное)

Бот для системы *Discord*

Презентация

					ПГУ1.090301.11.001 ПЗ	Лист
						64
Изм.	Лист	№ докум.	Подпись	Дата		

Тема работы: Бот для системы Discord

авторы: студент группы 19ВВП1 Кубасов И.М.
и руководитель: к.т.н., доцент каф. ВТ ПГУ Слепцов Н.

Пенза 2023

					ПГУ1.090301.11.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		65

Польность работы

бщения и взаимодействия между пользователями Discord стали невероятно популярными в сообществе. Они создают коммуникационное пространство, где пользователи могут собираться, обмениваться информацией и взаимодействовать друг с другом. Однако, с ростом популярности Discord возникают потребности в эффективном управлении и управлении ими.

Предмет и объект работы

ии реализации бота для системы Discord

и работы является реализация бота для
трирования Discord-сервера

					ПГУ1.090301.11.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		67

работы и постановка задач

и разработка и реализация бота для Discord-сервера, который будет функциями, направленными на упрощение процесса администрирования сообщества пользователям.

и

печить следующие функции:

ование;

ени в голосовых каналах;

звленных сообщений;

в опыта" и ролей;

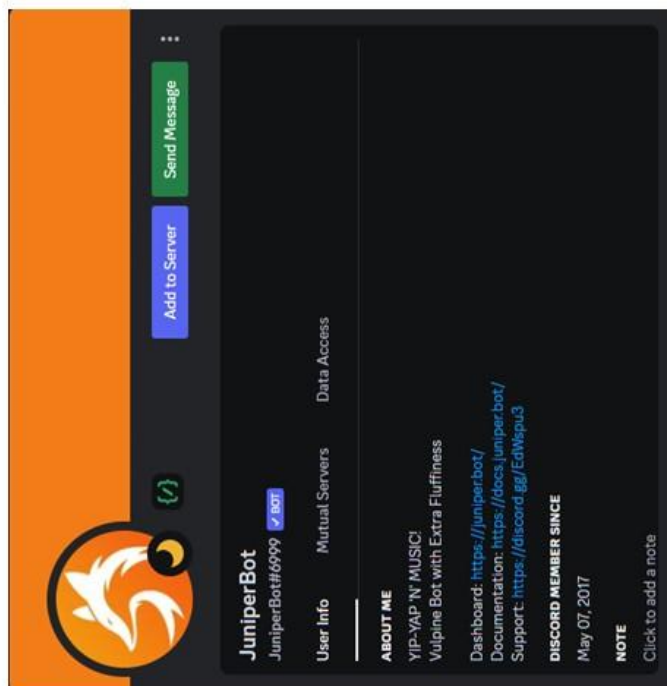
стики;

юкировка;

ый фильтр нецензурной лексики.

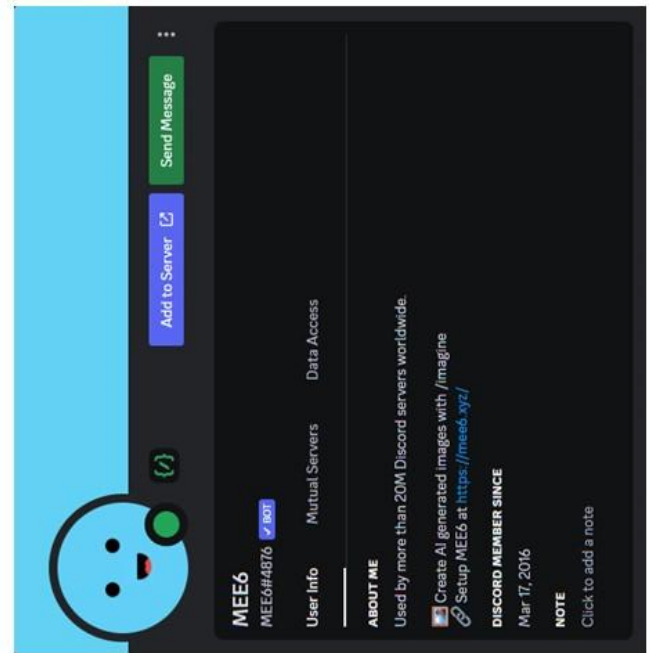
					ПГУ1.090301.11.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		68

Аналоги



JuniperBot

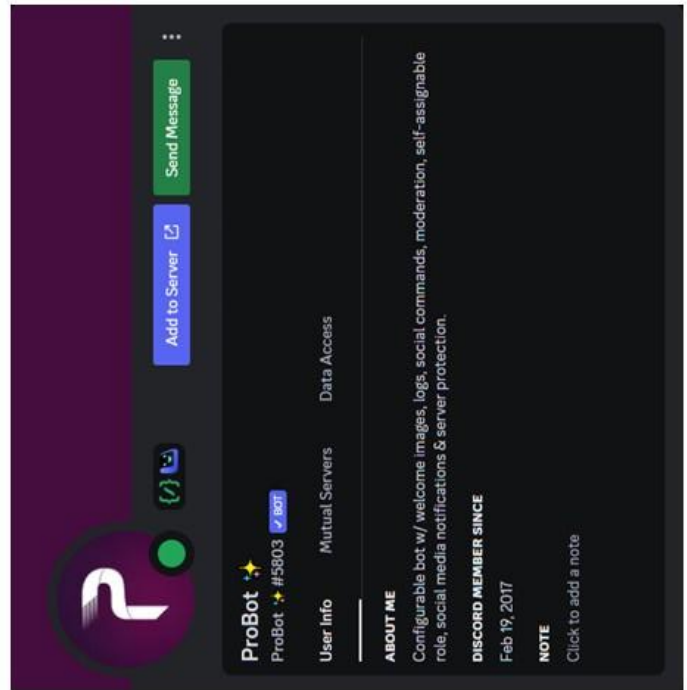
Аналоги



MEE6

					ПГУ1.090301.11.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		70

Аналоги



ProBot

Аналоги



Dyno

Стек технологий



Disc



					ПГУ1.090301.11.001 ПЗ	Лист
						73
Изм.	Лист	№ докум.	Подпись	Дата		

Диаграмма развертывания

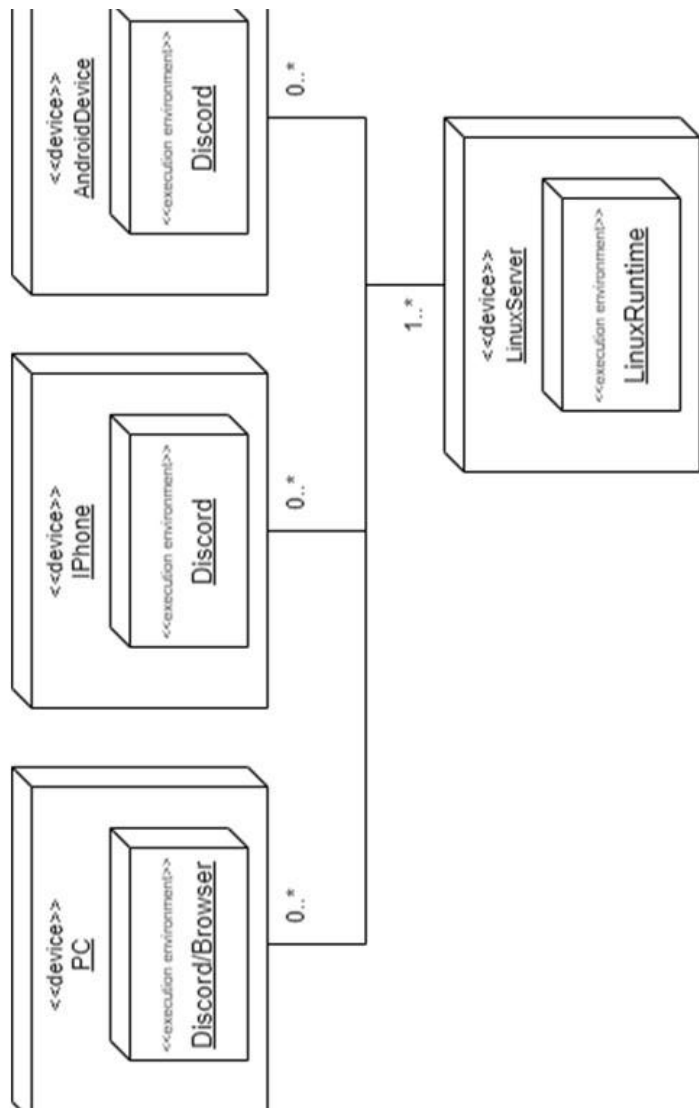


Диаграмма прецедентов

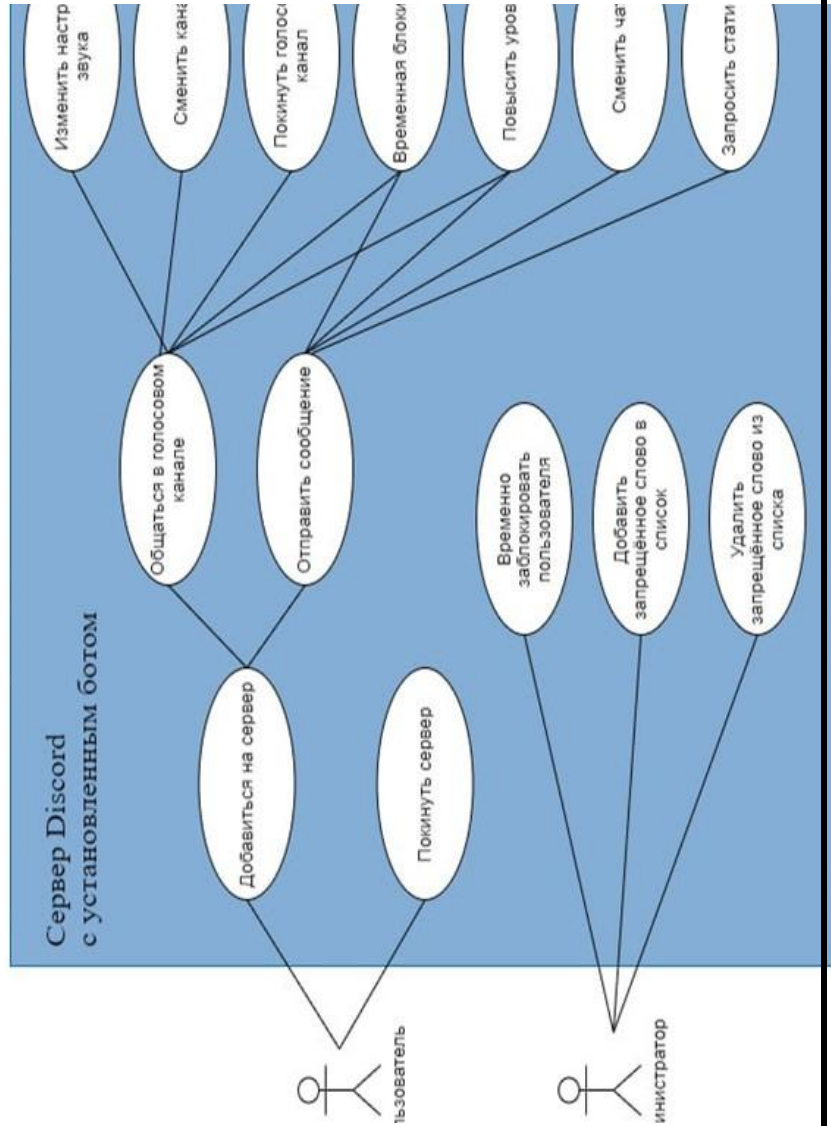
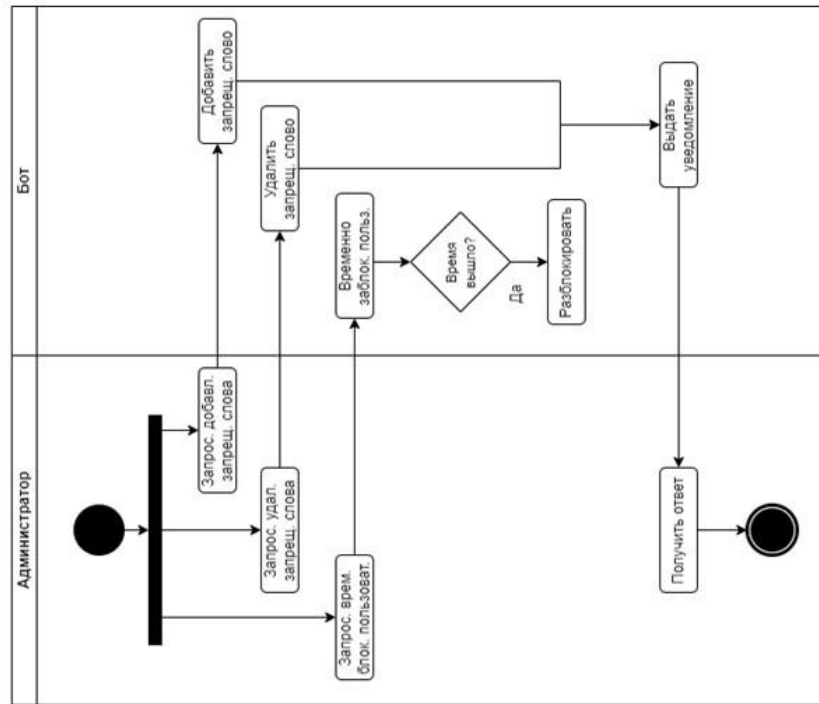
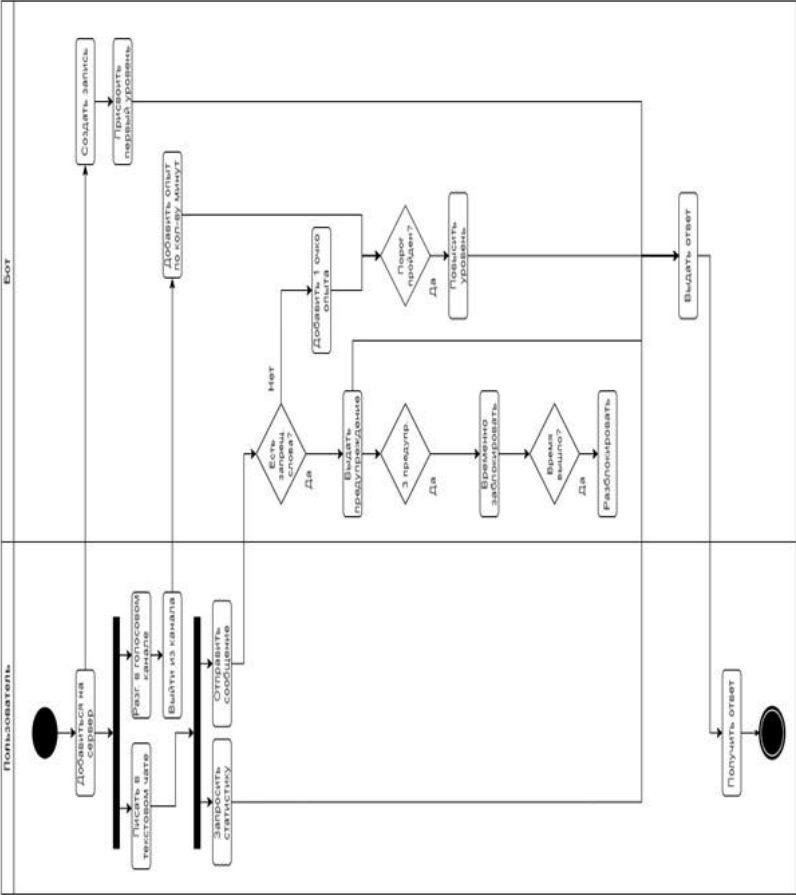


Диаграмма деятельности



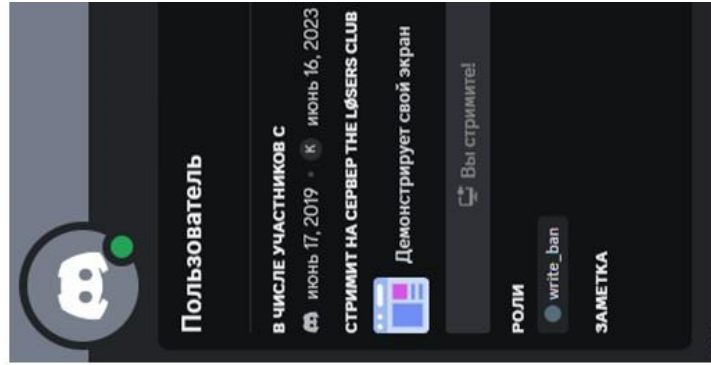
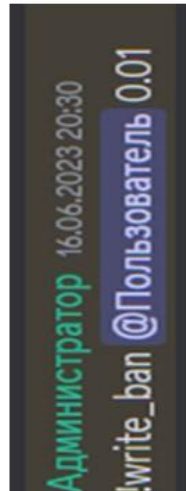
Для администратора

Диаграмма деятельности

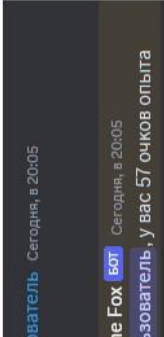
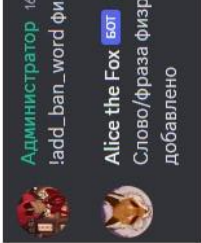
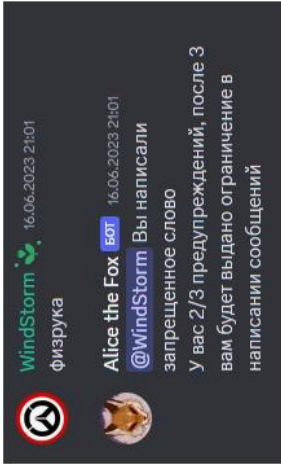


Для пользователя

Тестирование и отладка



Тестирование и отладка



Заключение

выполнения выпускной квалификационной работы бот для системы Discord, с использованием как основного языка программирования. Был использован библиотеки для логического анализа руторhу2.

Бот облегчает администрирование на сервере, сохраняет дружественную и безопасную атмосферу из достоинств реализованного бота является наличие фильтра слов и наличие выдачи системы уроков

асибо за вниман

					ПГУ1.090301.11.001 ПЗ	Лист
						81
Изм.	Лист	№ докум.	Подпись	Дата		

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

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

«Утверждаю»

Заведующий кафедрой

_____ М.А. Митрохин

« 26 » _____ апреля 20 23 г.

**ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ**

1. Студент Кубасов Иван Михайлович гр 19ВВП1 факультета вычислительной техники.

2. Тема работы Бот для системы Discord

Тема утверждена приказом ПГУ № 458/о от "26" апреля 2023 г.

3. Руководитель бакалаврской работы к.т.н., доцент, каф. ВТ ПГУ Слепцов Н.В.

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

Реализовать бота для системы Discord, помогающего в администрировании сервера. Реализовать настраиваемый фильтр слов, систему поощрения и временную блокировку пользователей. Провести тестирование и отладку каждого элемента приложения.

5. Перечень подлежащих разработке вопросов

Требуется произвести анализ предметных областей: изучить существующих ботов, выделить их достоинства и недостатки. Выбрать язык программирования, который подойдет для написания логики приложения. Определить архитектуру приложения и вспомогательные средства разработки.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Изучение предметной области	50	26.04.2023-30.04.2023	
2.	Анализ требований	10	01.05.2023-02.05.2023	
3.	Выбор технологий, применяемых при проектировании и разработке	50	03.05.2023-07.05.2023	
4.	Описание алгоритмов	40	08.05.2023-11.05.2023	
5.	Реализация бота для системы Discord	150	12.05.2023-29.05.2023	
6.	Оформление пояснительной записки	24	30.05.2023-31.05.2023	

Дата выдачи задания "26" апреля 2023 г.

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

Задание к исполнению принял студент _____ Кубасов И.М.
(подпись, дата)

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