**ПЕРЕЧЕНЬ УСЛОВНЫХ ОБОЗНАЧЕНИЙ**

**ВВЕДЕНИЕ**

Квантовые вычисления — это быстро развивающаяся область, которая способна произвести революцию в нашем представлении о вычислениях. Хотя перспективы этой технологии невероятны, разработка квантового оборудования все еще находится на ранней стадии, а затраты, связанные с созданием и обслуживанием квантовых компьютеров, астрономически высоки. В результате исследователи и разработчики изучают альтернативные решения для моделирования квантовых вычислений с целью сделать эту технологию более доступной и недорогой.

Целью данной дипломной работы является проектирование и разработка эмулятора квантовых вычислений, который может имитировать сложное и запутанное поведение квантовых компьютеров. Этот эмулятор будет точно представлять квантовые алгоритмы, квантовые затворы и квантовые схемы, предоставляя исследователям и разработчикам мощный инструмент для тестирования и оптимизации приложений квантовых вычислений.

Для достижения этой цели дипломная работа будет сосредоточена на трех ключевых задачах:

1. Провести всесторонний обзор существующей литературы по эмуляторам квантовых вычислений для выявления лучших методов разработки точного и эффективного эмулятора. Для этого необходимо проанализировать широкий спектр научных работ, академических журналов и технических отчетов, чтобы выявить наиболее эффективные подходы к эмуляции квантовых вычислений.
2. Разработать и внедрить надежный и прочный эмулятор квантовых вычислений, который может моделировать широкий спектр квантовых алгоритмов, квантовых ворот и квантовых схем.
3. Оценить производительность эмулятора и сравнить его с существующими эмуляторами и реальным квантовым оборудованием. Это потребует тщательного тестирования и проверки, чтобы убедиться, что эмулятор точно моделирует поведение квантовых вычислений в широком диапазоне условий.

Достижение этих целей позволит внести ценный вклад в область квантовых вычислений, предоставив исследователям и разработчикам инструмент для тестирования и оптимизации квантовых алгоритмов и схем. Это, в свою очередь, поможет ускорить развитие этой захватывающей и преобразующей технологии, сделав ее более доступной и приемлемой для публики.

# **1. ПОСТАНОВКА ЗАДАЧИ**

## **1.1. Обзор аналогов**

* IBM Qiskit – это популярная платформа квантовых вычислений с открытым исходным кодом, которая включает в себя квантовый эмулятор. Эмулятор может моделировать до 32 кубитов и может быть запущен на локальной машине или в облаке с помощью платформы квантовых вычислений IBM. Qiskit также включает в себя множество инструментов для проектирования и запуска квантовых алгоритмов, что делает его комплексным решением для исследователей и разработчиков.
* Google Cirq – еще одна популярная платформа квантовых вычислений с открытым исходным кодом, которая включает в себя квантовый эмулятор. Эмулятор может моделировать до 30 кубитов и может быть запущен на локальной машине или в облаке с помощью платформы квантовых вычислений Google. Cirq спроектирован так, чтобы быть очень настраиваемым, что делает его хорошим выбором для исследователей, которым необходим тонкий контроль над квантовыми симуляциями.
* Microsoft Q# и Quantum Development Kit: Microsoft Q# – это язык программирования для квантовых вычислений, который включает в себя квантовый эмулятор. Эмулятор может моделировать до 32 кубитов и может быть запущен на локальной машине или в облаке с помощью платформы квантовых вычислений Microsoft. Quantum Development Kit также включает инструменты для разработки и тестирования квантовых алгоритмов.
* Rigetti Forest – это облачная платформа квантовых вычислений, которая включает в себя квантовый эмулятор. Эмулятор может моделировать до 36 кубитов и спроектирован как масштабируемый, что позволяет пользователям моделировать более крупные квантовые системы по мере необходимости. Forest также включает в себя различные инструменты для проектирования и запуска квантовых алгоритмов, что делает его комплексным решением для исследователей и разработчиков.
* Xanadu Strawberry Fields – это платформа для квантовых вычислений, включающая квантовый эмулятор. Эмулятор может моделировать до 50 кубитов и разработан как очень настраиваемый, что позволяет пользователям моделировать широкий спектр квантовых систем. Strawberry Fields также включает в себя множество инструментов для разработки и тестирования квантовых алгоритмов.

В целом, эти квантовые эмуляторы являются мощными инструментами для исследователей и разработчиков, работающих в области квантовых вычислений. Они предлагают целый ряд функций и возможностей, от настраиваемых симуляторов до комплексных сред разработки, позволяя разрабатывать и тестировать квантовые алгоритмы без доступа к физическому квантовому оборудованию. Поскольку область квантовых вычислений продолжает развиваться, вполне вероятно, что эти эмуляторы станут еще более мощными и доступными, прокладывая путь к новым открытиям и инновациям в этой области.

## **1.2. SWOT-анализ**

SWOT-анализ — это схема для оценки сильных и слабых сторон, возможностей и угроз продукта или услуги.

|  |  |
| --- | --- |
| **Сильные стороны:**   * Точное моделирование: квантовые эмуляторы могут моделировать поведение сложных квантовых систем с высокой точностью, позволяя изучать и тестировать квантовые алгоритмы без доступа к физическому квантовому оборудованию. * Доступность: квантовые эмуляторы могут работать на широком спектре оборудования и часто доступны как инструменты с открытым исходным кодом, что делает их доступными для исследователей и разработчиков, у которых может не быть доступа к специальному квантовому оборудованию. * Настраиваемость: многие квантовые эмуляторы предлагают высокую степень настройки, позволяя исследователям точно настроить моделирование в соответствии с их конкретными исследовательскими потребностями. | **Слабые стороны:**   * Ограниченные возможности: квантовые эмуляторы ограничены вычислительной мощностью классических компьютеров и могут быть не в состоянии точно моделировать большие или сложные квантовые системы. * Отсутствие физического взаимодействия:   квантовые эмуляторы не позволяют пользователям наблюдать физическое поведение квантовых систем, что может ограничить их полезность для определенных типов исследований.   * Сложность: квантовые эмуляторы могут быть сложными и трудными в использовании, особенно для пользователей, которые являются новичками в области квантовых вычислений. |
| **Возможности:**   * Разработка новых алгоритмов: квантовые эмуляторы можно использовать для разработки и тестирования новых квантовых алгоритмов, что потенциально может привести к новым открытиям и инновациям в области квантовых вычислений. * Повышение доступности: поскольку квантовые эмуляторы становятся более мощными и простыми в использовании, они могут стать более доступными для более широкого круга исследователей и разработчиков, что приведет к расширению сотрудничества и инноваций в этой области. | **Угрозы:**   * Конкуренция со стороны физического квантового оборудования: по мере того как физическое квантовое оборудование становится более доступным, полезность квантовых эмуляторов может снизиться. * Риски безопасности: ожидается, что квантовые вычисления окажут сильное разрушительное воздействие на область кибербезопасности, а разработка мощных квантовых эмуляторов может потенциально увеличить риск атак на основе квантовых технологий. * Недостаток финансирования: без надлежащего финансирования и поддержки разработка квантовых эмуляторов может замедлиться, что ограничит их потенциал для будущих исследований и инноваций. |

Расписать ниже таблицы внутри кратко

## **1.3. PEST-анализ**

PEST-анализ – это основа для оценки политических, экономических, социальных и технологических факторов, которые могут повлиять на продукт или услугу. Вот PEST-анализ квантовых эмуляторов:

* Политические:

Государственное финансирование: Государственное финансирование может играть значительную роль в развитии квантовых эмуляторов, особенно в странах, которые вкладывают значительные средства в квантовые исследования и разработки.

Нормативно-правовая среда: Нормативные акты, касающиеся квантовых вычислений и квантового шифрования, могут потенциально повлиять на разработку и использование квантовых эмуляторов.

* Экономика:

Рыночный спрос: Спрос на квантовые эмуляторы может возрасти по мере того, как квантовые вычисления станут более широко применяться в таких отраслях, как финансы, здравоохранение и кибербезопасность.

Конкуренция: Появление новых игроков на рынке квантовых вычислений может привести к усилению конкуренции в области разработки и распространения квантовых эмуляторов.

* Социальные:

Общественная осведомленность: По мере того, как квантовые вычисления будут все более широко обсуждаться в средствах массовой информации, осведомленность общественности о квантовых эмуляторах и их потенциальном использовании может возрасти.

Этические проблемы: Разработка мощных квантовых эмуляторов может вызвать этические проблемы, связанные с возможностью атак на основе квантовых технологий или использования квантовых вычислений для наблюдения.

* Технологические:

Достижения в области классических вычислений: Разработка более мощных классических компьютеров может потенциально улучшить точность и масштабируемость квантовых эмуляторов.

Достижения в квантовом оборудовании: По мере дальнейшего совершенствования физического квантового оборудования полезность квантовых эмуляторов может уменьшиться. Однако квантовые эмуляторы все еще могут играть определенную роль в разработке и тестировании новых квантовых алгоритмов.

В целом, на развитие и использование квантовых эмуляторов влияет целый ряд политических, экономических, социальных и технологических факторов. По мере дальнейшего развития области квантовых вычислений эти факторы, вероятно, будут играть все более важную роль в формировании процесса разработки и внедрения квантовых эмуляторов.

**1.4. Формирование требований к программному продукту**

### **1.4.1. Бизнес требования**

Бизнес-цель – это главный фактор, необходимый для выполнения и реализации проекта. Главная цель связана с глобальными задачами и одновременно с самим проектом. В этом случае цель, предприниматель видит своей главной задачей просто создать продукт.

* Экономическая эффективность: Эмулятор должен быть экономически эффективным и предлагать конкурентоспособную цену по сравнению с другими решениями для квантовых вычислений.
* Гибкость: Эмулятор должен быть гибким и способным адаптироваться к изменяющимся потребностям рынка и развивающейся области квантовых вычислений.
* Масштабируемость: Эмулятор должен быть масштабируемым и способным удовлетворить потребности как малого, так и крупного бизнеса.
* Возможности партнерства: Эмулятор должен быть открыт для партнерства с другими компаниями и организациями в области квантовых вычислений.
* Получение доходов: Эмулятор должен иметь потенциал для получения дохода за счет лицензирования, партнерства или других бизнес-моделей.

### **1.4.2. Пользовательские требования**

У пользователя должны быть следующие возможности:

* Простота использования: Эмулятор должен быть простым в использовании, с хорошо продуманным интерфейсом и понятной документацией, даже для пользователей, которые являются новичками в области квантовых вычислений.
* Точность: Эмулятор должен быть способен выдавать точные результаты, соответствующие поведению физических квантовых систем.
* Скорость: эмулятор должен быть способен выполнять вычисления быстро и эффективно, не будучи ограниченным вычислительной мощностью классического оборудования.
* Настраиваемость: Эмулятор должен быть настраиваемым, позволяя пользователям точно настраивать моделирование в соответствии с их конкретными исследовательскими потребностями.
* Поддержка: Эмулятор должен иметь сильную систему поддержки, включая документацию, техническую поддержку и учебные материалы.
* Совместимость: Эмулятор должен быть совместим с целым рядом классических вычислительных аппаратных и программных платформ, что позволит пользователям интегрировать эмулятор в существующие рабочие процессы.

### **1.4.3. Функциональные требования**

Функциональные требования – определяют функциональность (поведение) программной системы, которая должная быть создана разработчиками для предоставления возможности выполнения пользователями своих обязанностей в рамках бизнес-требований и в контексте пользовательских требований.

В ходе разработки проекта были определены следующие функциональные требования:

* Точные квантовые вычисления: Эмулятор должен быть способен выполнять точные квантовые вычисления, которые соответствуют поведению физических квантовых систем.
* Поддержка нескольких квантовых алгоритмов: Эмулятор должен поддерживать ряд квантовых алгоритмов, включая те, которые используются для коррекции ошибок, оптимизации и моделирования.
* Масштабируемость: Эмулятор должен быть масштабируемым и способным моделировать большие и более сложные квантовые системы по мере необходимости.
* Настраиваемость: Эмулятор должен быть настраиваемым, позволяя исследователям и разработчикам точно настраивать моделирование в соответствии с их конкретными исследовательскими потребностями.
* Совместимость: Эмулятор должен быть совместим с целым рядом классических вычислительных аппаратных и программных платформ.

### **1.4.4. Нефункциональные требования**

Нефункциональные требования – требования, определяющие свойства, которые система должна демонстрировать, или ограничения, которые она должна соблюдать, не относящиеся к поведению системы.

Нефункциональными требованиями для данного программного продукта являются:

* Производительность: Эмулятор должен быть способен выполнять вычисления с высокой скоростью и точностью, не будучи ограниченным вычислительной мощностью классического оборудования.
* Надежность: Эмулятор должен быть надежным и способным давать стабильные результаты даже при моделировании сложных квантовых систем.
* Безопасность: Эмулятор должен быть безопасным и защищать конфиденциальные данные от несанкционированного доступа или злонамеренных атак.
* Удобство использования: Эмулятор должен быть удобным и простым в использовании, с хорошо продуманным интерфейсом и понятной документацией.
* Доступность: Эмулятор должен быть доступен для исследователей и разработчиков, независимо от их уровня знаний или доступа к физическому квантовому оборудованию.
* Совместимость со стандартами: Эмулятор должен соответствовать промышленным стандартам квантовых вычислений и моделирования, обеспечивая совместимость с другими инструментами и платформами квантовых вычислений.

### **1.4.5. Ограничения**

* Масштабируемость: хотя квантовые эмуляторы могут моделировать более крупные квантовые системы, чем те, которые в настоящее время возможны с помощью физических квантовых компьютеров, все еще существуют ограничения на размер и сложность систем, которые могут быть точно смоделированы.
* Вычислительные ресурсы: Вычислительные ресурсы, необходимые для работы квантовых эмуляторов, могут быть значительными, особенно для больших и сложных систем, что может ограничить их доступность и дешевизну.
* Точность: хотя квантовые эмуляторы нацелены на получение точных результатов, всегда существует риск ошибок и неточностей, особенно по мере увеличения размера и сложности моделируемой системы.
* Ограниченная область применения: Квантовые эмуляторы могут моделировать только квантовые системы и не могут быть использованы для классических вычислительных задач.

### **1.4.6. Требования к интерфейсам**

* Удобный интерфейс: Эмулятор должен иметь дружественный интерфейс, в котором легко ориентироваться и который понятен даже пользователям, впервые пришедшим в область квантовых вычислений.
* Инструменты визуализации: Эмулятор должен предоставлять инструменты визуализации, которые позволят пользователям лучше понять результаты моделирования и получить представление о поведении моделируемых квантовых систем.
* Форматы ввода/вывода: Эмулятор должен поддерживать ряд форматов ввода/вывода, что облегчает интеграцию эмулятора в существующие рабочие процессы и программные инструменты.
* Языки программирования: Эмулятор должен поддерживать ряд языков программирования, позволяя пользователям писать код на том языке, который им наиболее удобен.
* Документация: Эмулятор должен иметь четкую и полную документацию, включая руководства пользователя, учебные пособия и технические спецификации.

## **1.5. Программные средства разработки**

Для разработки использовалось следующие программные средства:

* Microsoft Visual Studio Community
* Язык программирования C#
* Windows Presentation Foundation
* Визуальный редактор сцен SceneBuilder для WPF

## **1.6. Аппаратные средства разработки**

Для разработки дипломного проекта был использован персональный компьютер со следующими характеристиками:

* Процессор – Intel(R) Core(TM) i5-8259U CPU @ 2.30GHz;
* Встроенная видеокарта – Intel(R) Iris(R) Plus Graphics 655;
* Оперативная память – 8 ГБ;
* Накопитель (SSD) – 256 ГБ;
* Операционная система – Windows 10.

# **2. ПРОЕКТИРОВАНИЕ ПРОГРАММНЫХ МОДУЛЕЙ**

После определения задач и функций программного продукта необходимо разработать структуру приложения и представить в виде UML диаграмм.

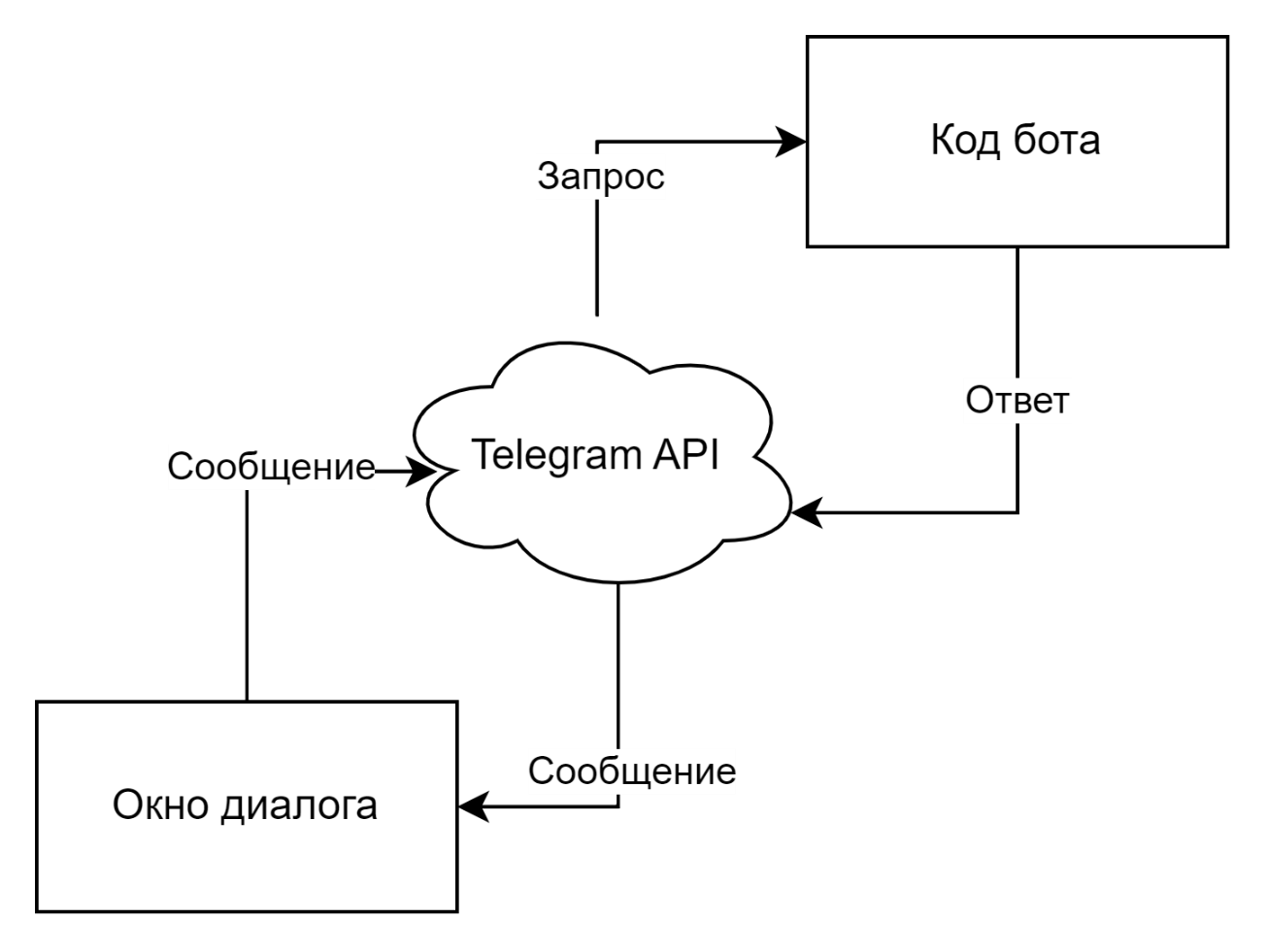
Процесс функционирования программы заключается в следующем. После начала работы пользователь может использовать указанные в боте команды и кнопки для взаимодействия и выполнения задач. Бот в ответ будет отправлять

Для моделирования основных сценариев разрабатываемого продукта был выбран сервис Draw.io – векторный графический редактор, редактор диаграмм и блок-схем.

## **2.1. Архитектура системы**

Архитектура системы — это принципиальная организация системы, воплощенная в её элементах, их взаимоотношениях друг с другом и со средой, а также принципы, направляющие её проектирование.

При проектирование программного продукта использовался TelegramBotAPI, обеспечивающий связь сообщений пользователя и запросов для бота.



*Рис. 2.1.1 – Архитектура системы*

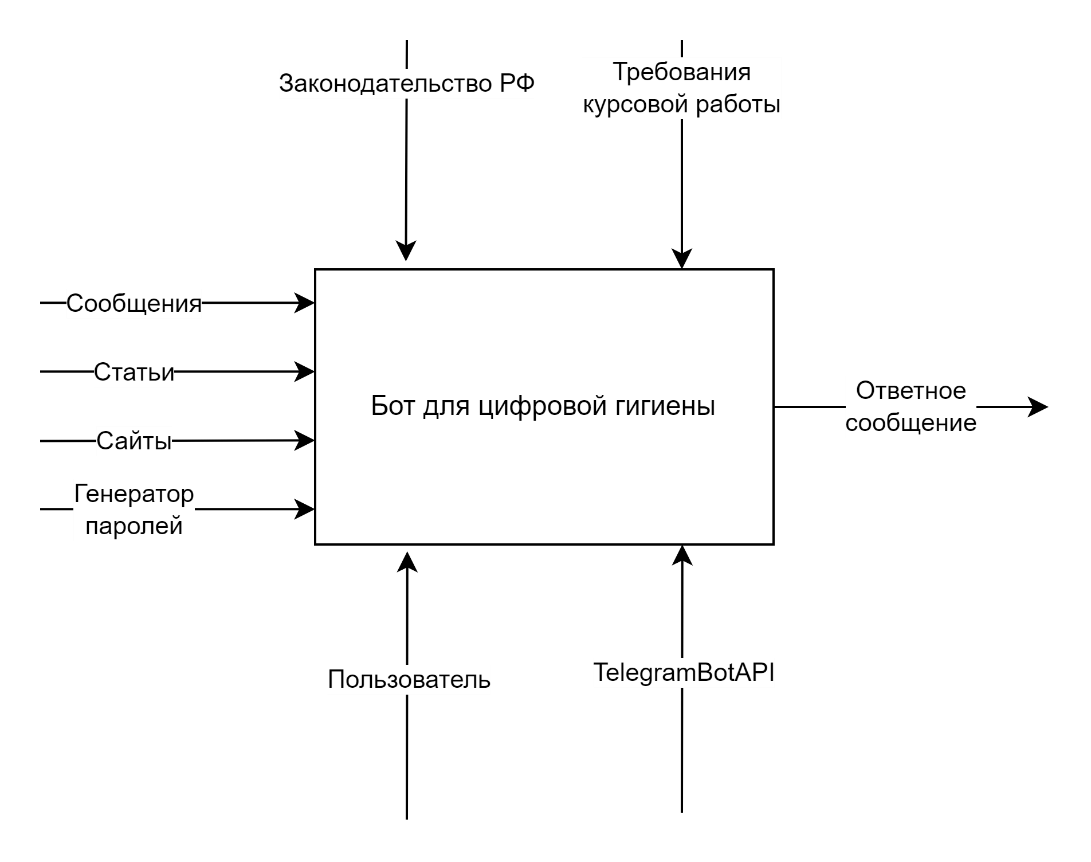
## **2.2. Моделирование основных сценариев системы**

Для моделирования основных сценариев системы был использован Draw.io и стандартные нотации IDEF0 и UML.

IDEF0 (Integration Definition Function Modeling) – это методология функционального моделирования и графическая нотация, предназначенная для визуализации и описания бизнес-процессов.

UML (Unified Modeling Language) — это стандартизированный язык моделирования, состоящим из интегрированного набора диаграмм, разработанных, чтобы помочь разработчикам систем и программного обеспечения в определении, визуализации, конструировании и документировании артефактов программных систем, а также для бизнес-моделирования. Моделирование основн ых сценариев систем с помощью диаграмм в нотации IDFE0 и UML позволяет облегчить коммуникацию как внутри команды разработчиков, так и при общении с заказчиком. Такие диаграммы достаточно просты для понимания.

Для того, чтобы визуализировать назначение системы, ее основную функцию, взаимодействие с внешней средой и описать ее границы была построена контекстная диаграмма в нотации IDEF0 (рис. 2.2.1):



*Рис. 2.2.1 Контекстная диаграмма*

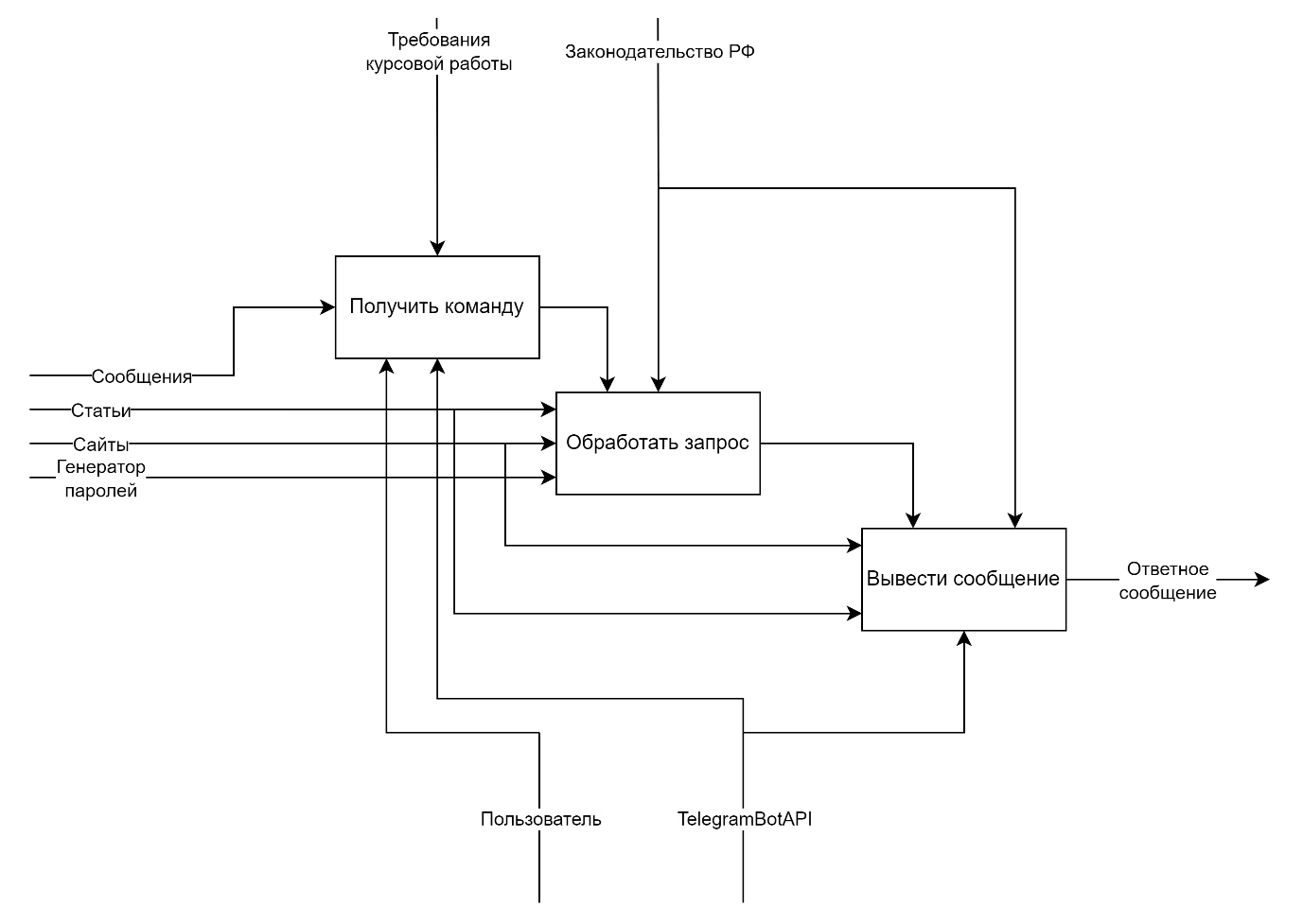
Для контекстной диаграммы составлено описание интерфейсных дуг процесса в виде таблицы (табл. 2.2.1):

Таблица 2.2.1

Словарь интерфейсных дуг процесса

|  |  |
| --- | --- |
| Имя | Описание |
| 1 | 2 |
| Сообщение | Сообщение от пользователя, являющееся запросом на выполнение действия ботом. |
| Ответное сообщение | Сообщение, отвечающее пользователю, содержащее запрошенное выполненное действие. |
| Статьи | Интернет статьи, содержащие полезную информацию |
| Сайты | Сайты с полезной информацией |
| Генератор паролей | Подпрограмма бота, создающая безопасные пароли. |
| TelegramBotAPI | Средство связи пользователя и кода бота, необходимо для выполнения запросов. |

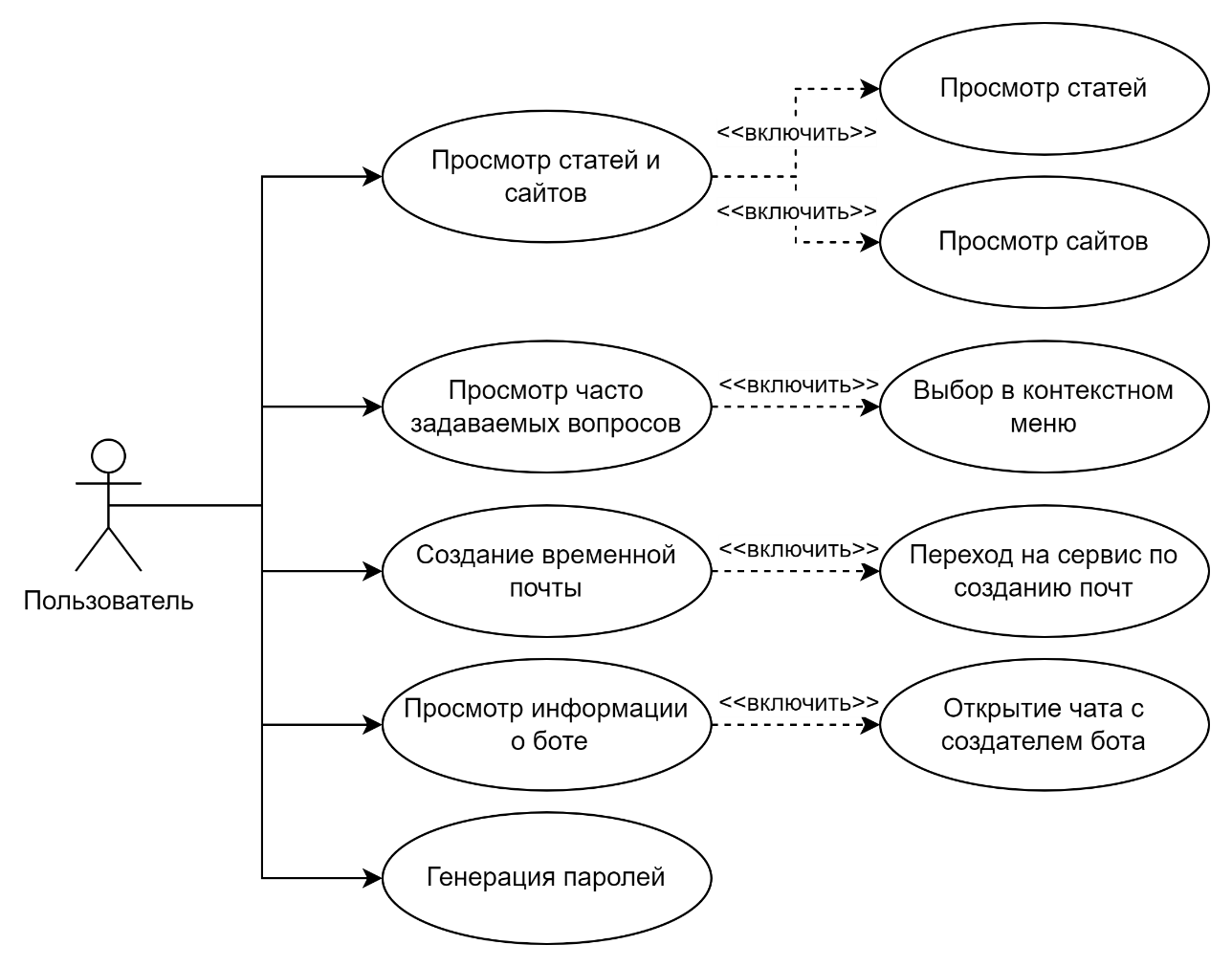
На основе контекстной диаграммы была построена диаграмма декомпозиции (рис. 2.2.2), на которой можно подробно увидеть выполняемые функции:



*Рис. 2.2.2 – Диаграмма декомпозиции*

Далее для определения общих границ и контекста моделируемой предметной области была построена диаграмма вариантов использования (use case).

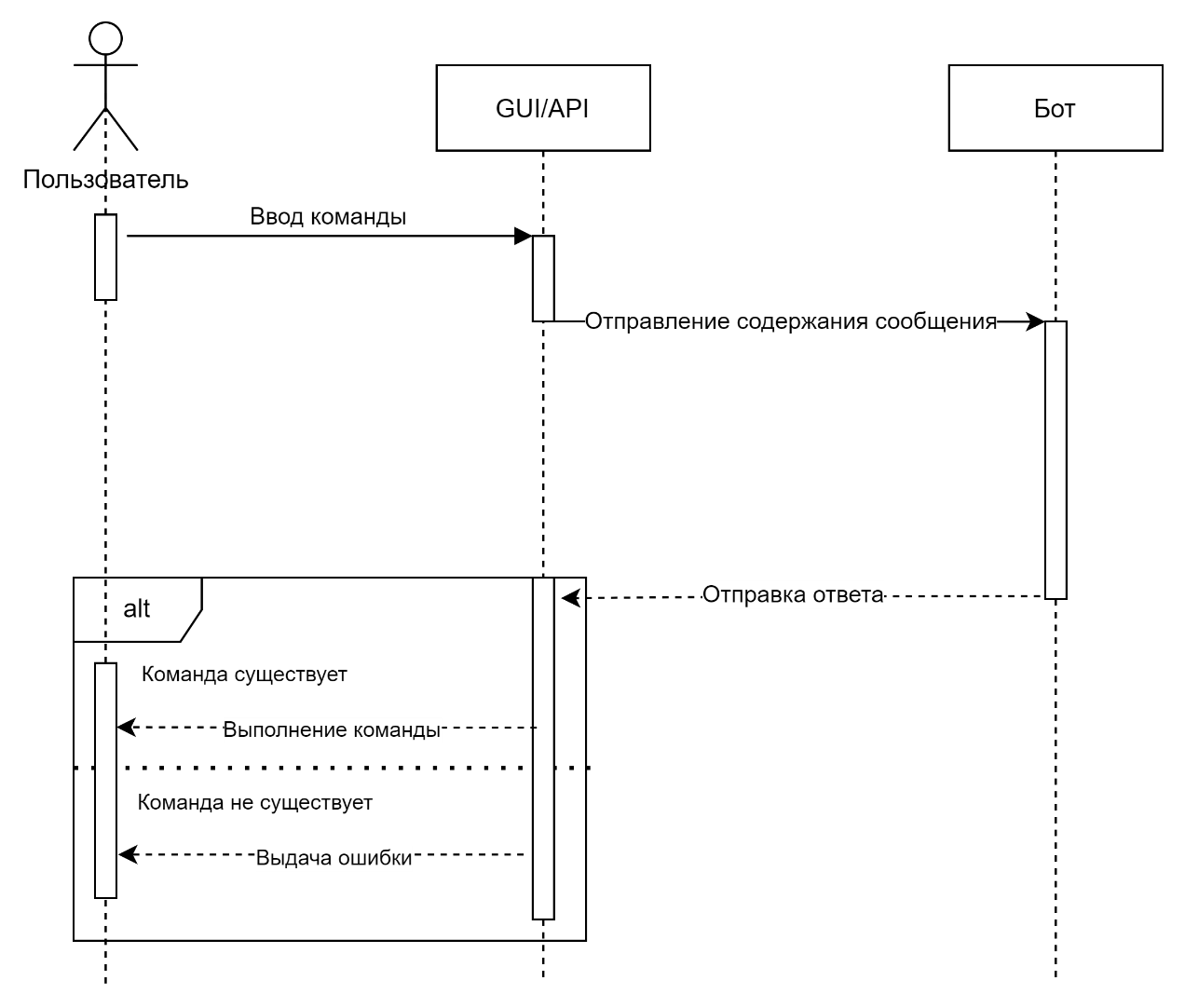
Диаграмма вариантов использования (use-case) в UML — диаграмма, отражающая отношения между актерами и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне. Построенная в рамках проектирования курсового проекта диаграмма вариантов использования для роли Пользователь представлена на рисунке 2.2.3:



*Рис. 2.2.3 – Диаграмма вариантов использования для Пользователя*

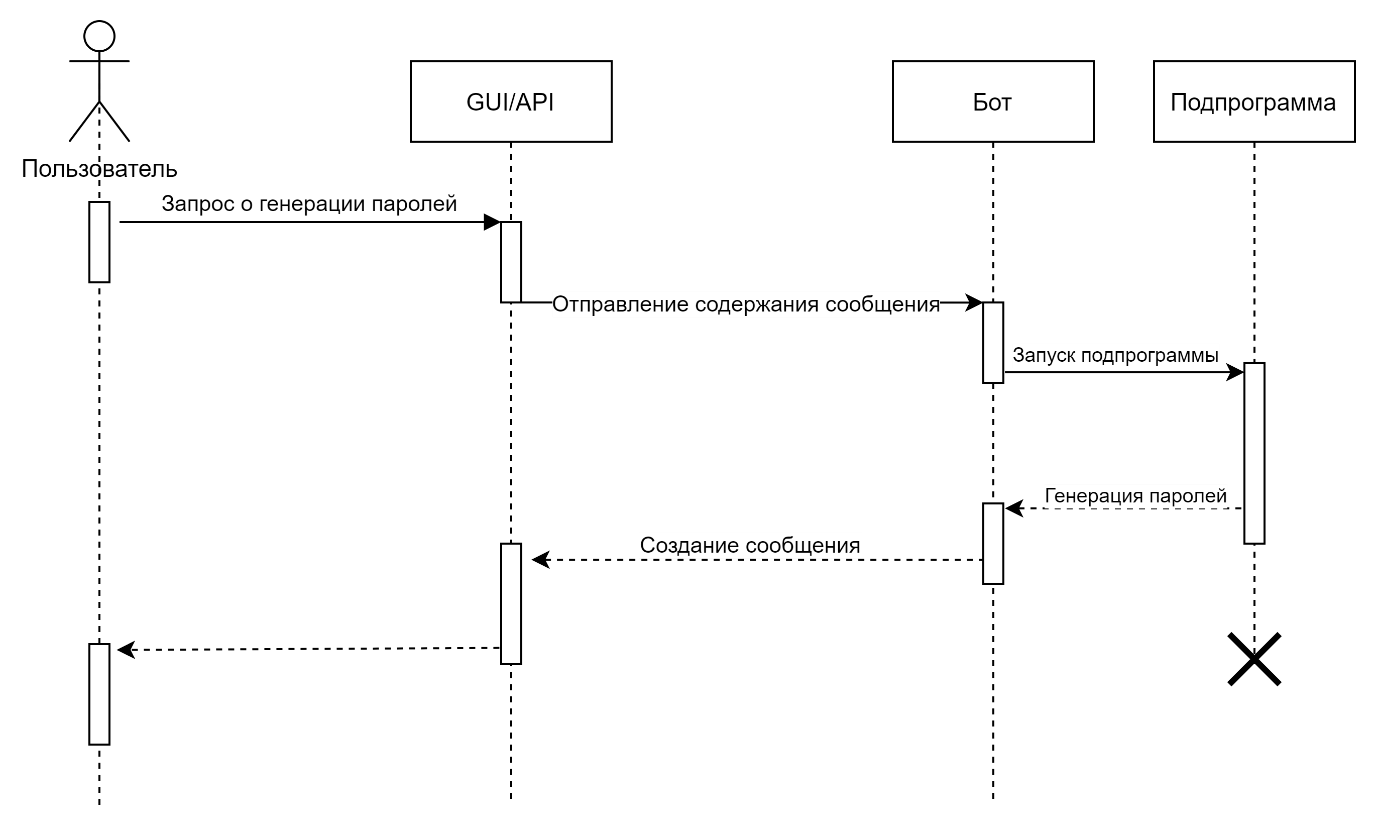
Диаграмма последовательности — UML-диаграмма, на которой для некоторого набора объектов на единой временной оси показан жизненный цикл объекта и взаимодействие актеров информационной системы в рамках прецедента.

Диаграмма последовательностей «Ввод команды» (рис. 2.2.4). На данной диаграмме отображен процесс ввода команды в бота.



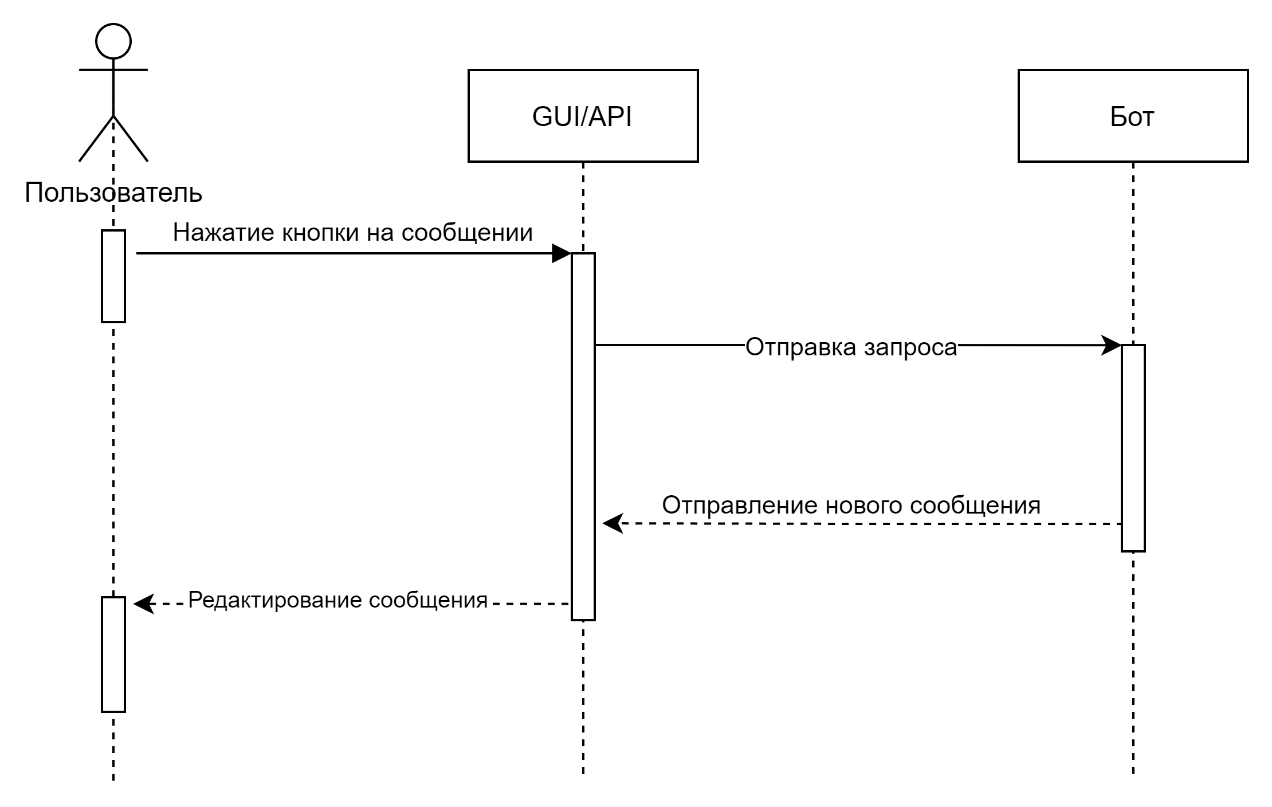
*Рис. 2.2.4 – Диаграмма последовательностей «Ввод команды»*

Диаграмма последовательностей «Запрос паролей» (рис. 2.2.5). На данной диаграмме отображен процесс запроса генерации безопасных паролей.



*Рис. 2.2.5 – Диаграмма последовательностей «Запрос паролей»*

Диаграмма последовательностей «*Нажатие кнопки на сообщении*» (рис. 2.2.6). На данной диаграмме отображен процесс нажатия интерактивной кнопки на сообщении от бота.

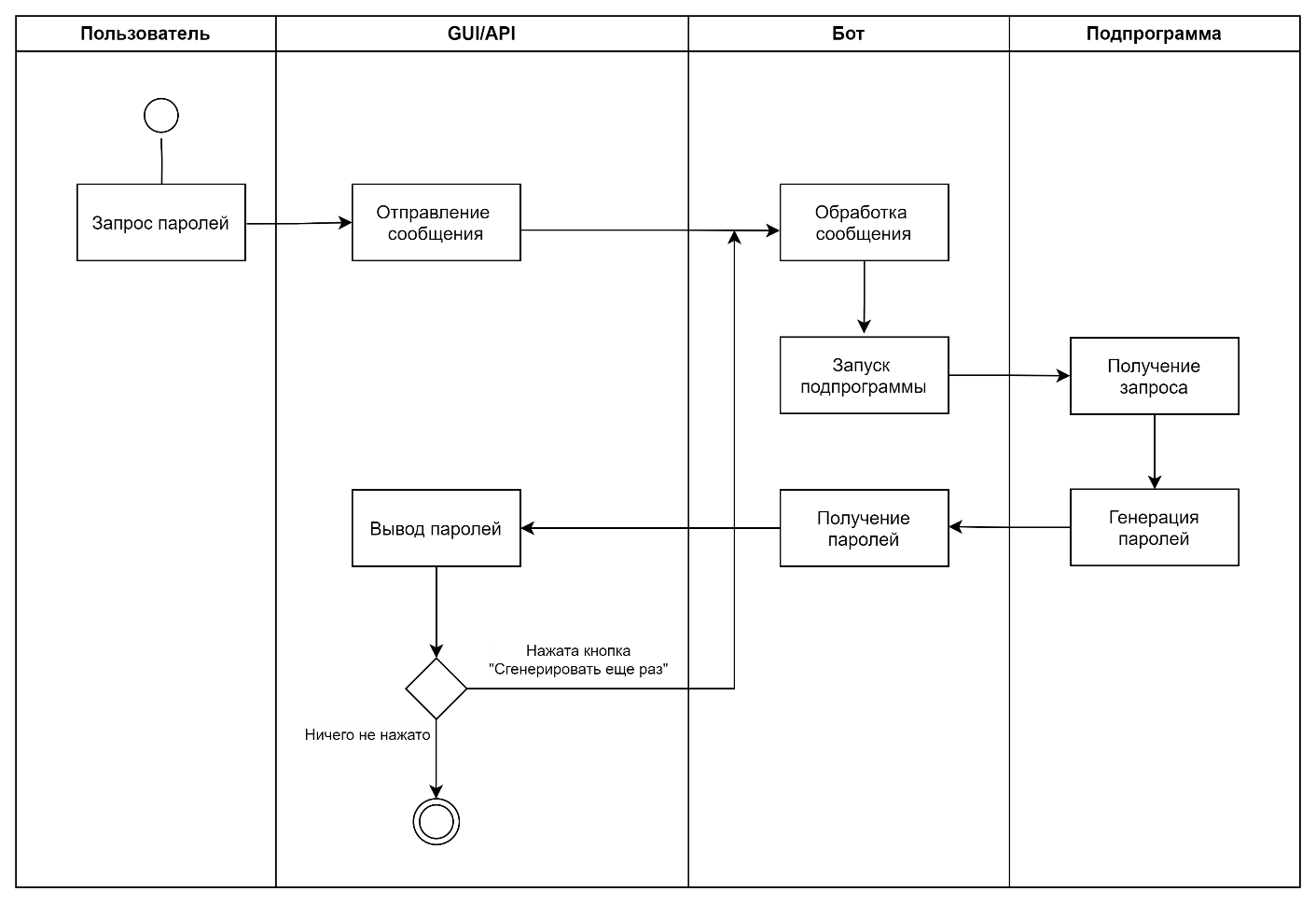


*Рис. 2.2.6 – Диаграмма последовательностей «Нажатие кнопки на сообщении»*

Следующие диаграммы, построенные в рамках дипломного проекта, – диаграммы деятельности (активности). Такие диаграммы описывают динамические аспекты поведения системы в виде блок-схемы, которая отражает бизнес-процессы, логику процедур и потоки работ — переходы от одной деятельности к другой.

Для разрабатываемого программного продукта были построены четыре диаграммы деятельности для событий «Запрос паролей», «Статьи», «Сайты», «Частые вопросы».

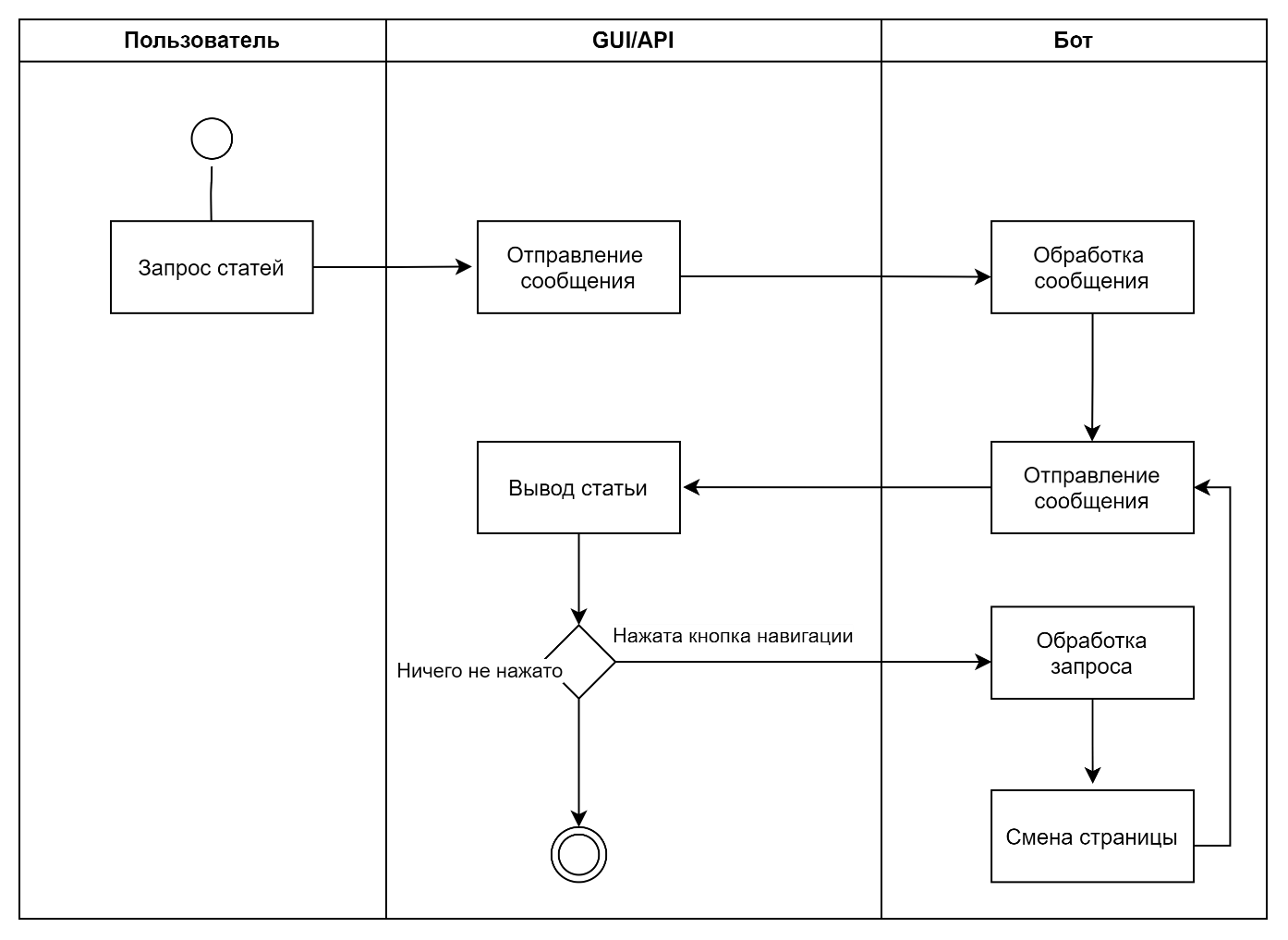
Диаграмма деятельности для события «Запрос паролей» (рис. 2.2.7):



*Рисунок. 2.2.7 – Диаграмма деятельности «Запрос паролей»*

Здесь видно как работает запрос паролей, отдельная генерация и цикл создания при нажатии кнопки на сообщении от бота.

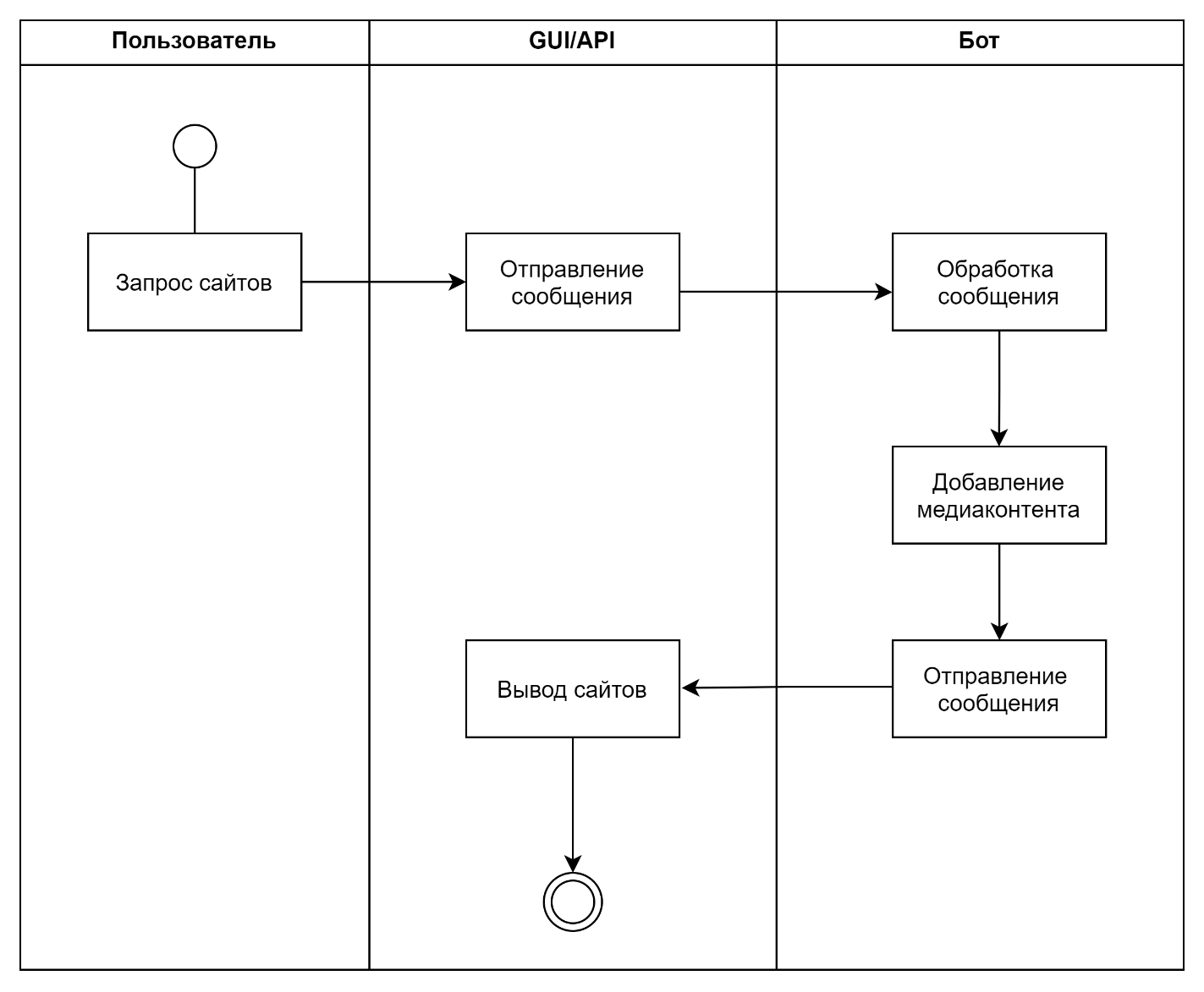
Следующая диаграмма деятельности для события «Статьи» (рис. 2.2.8):



*Рисунок 2.2.8 – Диаграмма деятельности «Статьи»*

На этой диаграмме видно, как работает запрос статей, а также нажатие кнопки навигации, находящейся на сообщении от бота.

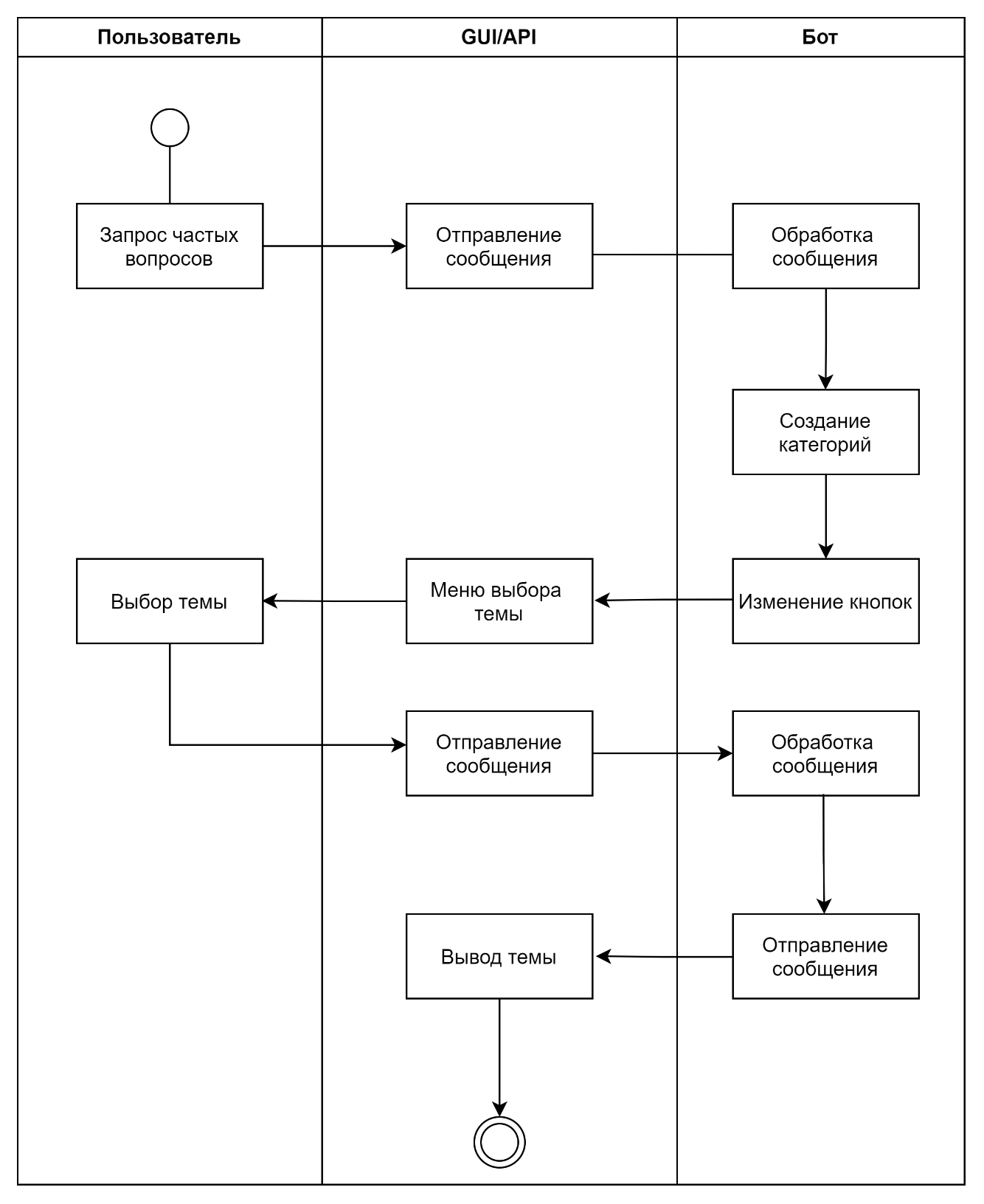
Диаграмма деятельности для события «Сайты» (рис. 2.2.9):



*Рисунок 2.2.9 – Диаграмма деятельности «Сайты»*

Здесь видно как работает запрос сайтов.

Диаграмма деятельности для события «Частые вопросы» (рис. 2.2.10):

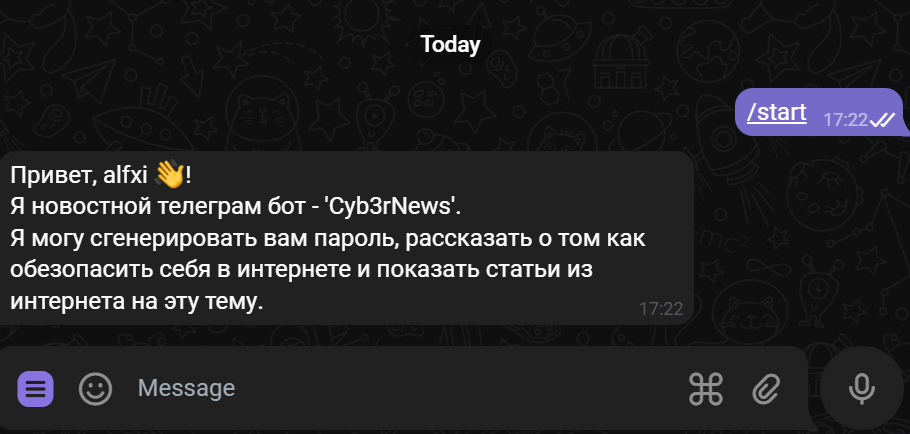


*Рисунок 2.2.10 – Диаграмма деятельности «Частые вопросы»*

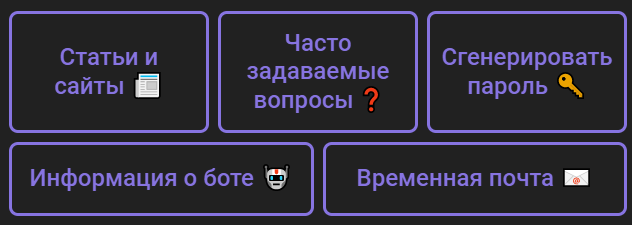
На этой диаграмме видно, как работает окно кнопок при выборе темы.

## **2.3. Проектирование графического интерфейса пользователя**

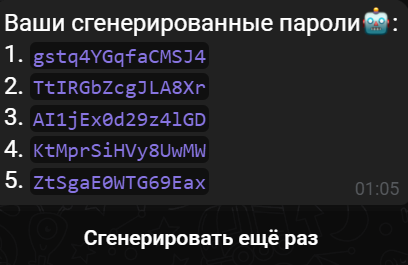
Графический интерфейс пользователя (GUI) — система средств для взаимодействия пользователя с компьютером, основанная на представлении всех доступных пользователю системных объектов и функций в виде графических компонентов экрана (окон, значков, меню, кнопок, списков и т. п.). В отличие от интерфейса командной строки, в GUI пользователь имеет произвольный доступ (с помощью устройств ввода — клавиатуры, мыши, джойстика и т. п.) ко всем видимым экранным объектам (элементам интерфейса) и осуществляет непосредственное манипулирование ими. Графический интерфейс пользователя должен быть интуитивно понятен, не запутан и в то же время не перегружен элементами.

В ходе проектирования графического интерфейса пользователя в рамках курсовой работы было построено дерево диалогов, использующее встроенные кнопки, и приветствие при начале работы: 

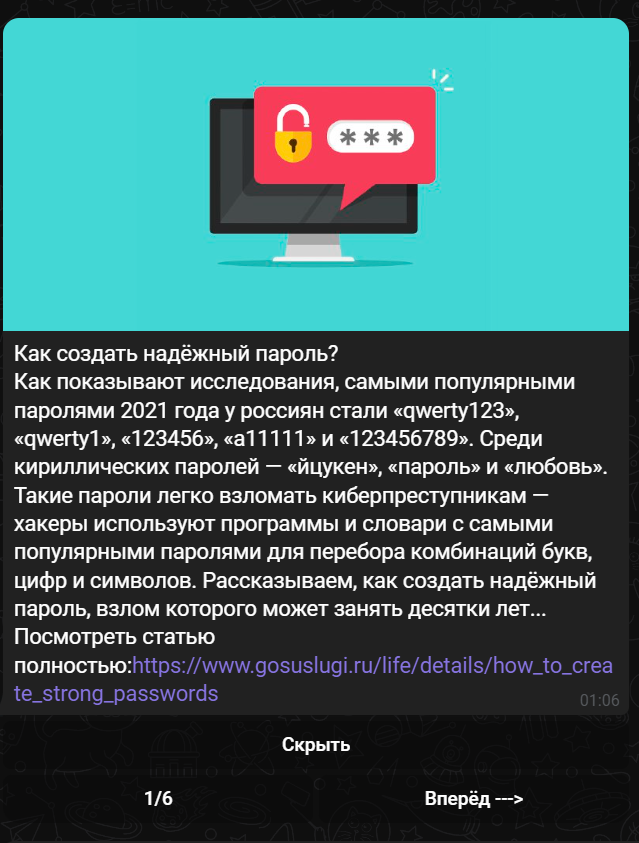
*Рис 2.3.1 Начало работы*



*Рис 2.3.2 Кнопки бота*



*Рис 2.3.3 Сообщение генерации паролей*



*Рис 2.3.4 Листание статей*

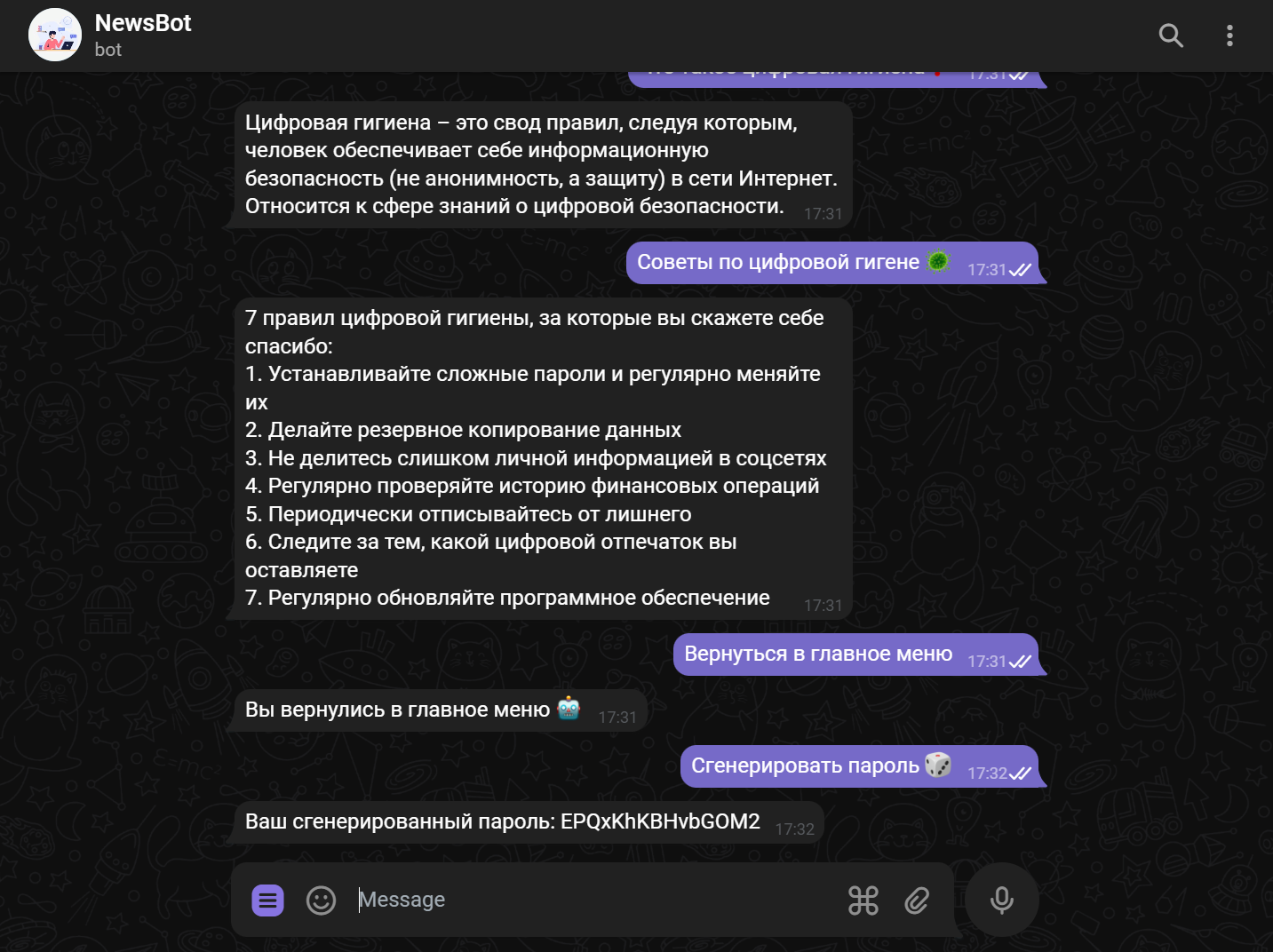
# **3. РАЗРАБОТКА ПРОГРАММНОГО ПРОДУКТА**

Для разработки telegram-бота был использован IDE PyCharm. это кроссплатформенная интегрированная среда разработки для языка программирования Python, разработанная компанией JetBrains на основе IntelliJ IDEA. Предоставляет пользователю комплекс средств для графической отладки и работы с кодом.

* В PyCharm легко редактировать код благодаря автодополнению, проверке кода на лету, подсветке ошибок и быстрым исправлениям, автоматическому рефакторингу и удобной навигации.
* Поддержка научных вычислений
* Помимо Python, PyCharm поддерживает JavaScript, CoffeeScript, TypeScript, Cython, SQL, HTML/CSS, языки шаблонов, AngularJS, Node.js и другие технологии.
* C PyCharm можно запускать, отлаживать, тестировать и развертывать приложения на удаленных хостах или виртуальных машинах с помощью удаленных интерпретаторов, встроенного SSH-терминала и интеграции c Docker и Vagrant.
* PyCharm предоставляет встроенный отладчик и инструмент запуска тестов, профилировщик Python, встроенный терминал, инструменты для работы с базами данных и интеграцию с популярными системами контроля версий.

## **3.1 Разработка графического интерфейса пользователя**

Разрабатываемый программный продукт использует внутренние средства приложения Telegram для отображения графического интерфейса, и будет выглядеть в основном так, как настроена тема приложения у пользователя. При изменении данных настроек вид разрабатываемого продукта также изменится, что создает огромное количество различных вариаций вида бота. Вот одно из них:

  
*Рис. 3.1 Графический интерфейс продукта*

**3.2 Разработка функционала**

Начало работы начинается с подключения TelergamBotAPI для обработки запросов.

Один из способов подключения – это ввод команды в терминал.

* pip install pyTelegramBotAPI

После этого необходимо подключить библиотеку telebot и можно начинать работу:

import telebot

Несколько библиотек и функций оттуда для упрощения работы:

import json

import string

from telebot import types # для указание типов

from telebot.types import InputMediaPhoto

from random import choice

Ввод ключа для доступа коду к боту.

bot = telebot.TeleBot("5808647633:AAFJJ4sgJyVo2NDAcyjI4L4As55nZ1mgEvw")

Photos - коллекция, содержащая путь (ссылки) на все фотографии

photos = [

'https://i.ibb.co/jRwKqh0/post1.jpg',

'https://i.ibb.co/t3gMTzp/post2.jpg',

'https://i.ibb.co/zhLWjgB/post3.png',

'https://i.ibb.co/hgfwrv8/post4.jpg',

'https://i.ibb.co/p4nGdyf/post5.png',

'https://i.ibb.co/RjJbQp0/post6.jpg']



*Рис. 3.2 Пример фотографии из коллекции*

Далее идет текст статей:

captions\_text = [

'Как создать надёжный пароль?\n

Как показывают исследования, самыми популярными паролями 2021 года у россиян стали «qwerty123», «qwerty1», «123456», «a11111» и «123456789». Среди кириллических паролей — «йцукен», «пароль» и «любовь». Такие пароли легко взломать киберпреступникам — хакеры используют программы и словари с самыми популярными паролями для перебора комбинаций букв, цифр и символов. Рассказываем, как создать надёжный пароль, взлом которого может занять десятки лет...\n

Посмотреть статью полностью:

https://www.gosuslugi.ru/life/details/how\_to\_create\_strong\_passwords',

'Как не стать жертвой фишинга?\n

Фишинг — вид интернет-мошенничества, цель которого получить доступ к секретным данным пользователя: логинам и паролям, номерам карт, банковским счетам. Преступники присылают фишинговые письма, которые могут быть очень похожи на настоящие сообщения от банков, компаний, органов власти или Госуслуг. Но ссылка в таком письме ведёт на поддельный сайт. Став жертвой фишинга, можно лишиться денег или доступа к своим аккаунтам, пустить хакера в корпоративную сеть работодателя. Фишинговыми бывают не только письма, приходящие на электронную почту. Это могут быть сообщения в мессенджерах, социальных сетях и смс. Рассказываем, как распознать и защититься от фишинга...\n

Посмотреть статью полностью:

https://www.gosuslugi.ru/life/details/dont\_get\_caught\_phishing',

'Как стать киберграмотным в 60 лет?\n

Интернетом активно пользуются не только молодёжь, но и пожилые люди. Если любите проводить время в соцсетях, переписываться в мессенджерах и не хотите рисковать своими накоплениями, изучите несколько простых советов для защиты от мошенников...\nПосмотреть статью полностью:

https://www.gosuslugi.ru/life/details/become\_cyber\_literate\_at\_60',

'Как защитить мобильное устройство?\n

Киберпреступники постоянно охотятся за чужими личными данными. Часто их целью является секретная информация из смартфонов: номера карт, доступы к онлайн-банкам, домашний адрес, рабочие документы и личные фото. Рассказываем, как защитить свой телефон от действий злоумышленников...\nПосмотреть статью полностью

:https://www.gosuslugi.ru/life/details/how\_to\_protect\_your\_mobile\_device',

'Как распознать звонок мошенника?\n

По данным МВД, за первое полугодие 2022 года телефонные мошенники похитили у россиян около 39 млрд ₽. Преступники постоянно придумывают новые способы обмана. Чтобы не попасться на их уловки, надо знать, как распознать мошенника и что делать при подобных звонках. Эти знания особенно важны для пожилых людей, поскольку именно они наиболее подвержены телефонному мошенничеству...\nПосмотреть статью полностью:

https://www.gosuslugi.ru/life/details/cyber\_security\_for\_kids?categoryCode=Internet\_and\_communication',

'Что рассказать детям о кибербезопасности?\n

Современные дети много времени проводят в интернете, поэтому важно научить их безопасному поведению в цифровом пространстве...\nПосмотреть статью полностью:

https://www.gosuslugi.ru/life/details/dont\_get\_caught\_phishing']

Начало работы бота (рис. 3.3, 3.4):

@bot.message\_handler(commands=['start'])

def start(message):

markup = types.ReplyKeyboardMarkup(resize\_keyboard=True)

btn1 = types.KeyboardButton("Статьи и сайты 📰")

btn2 = types.KeyboardButton("Часто задаваемые вопросы ❓")

btn3 = types.KeyboardButton("Сгенерировать пароль 🔑")

btn4 = types.KeyboardButton("Информация о боте 🤖")

btn5 = types.KeyboardButton("Временная почта 📧")

markup.add(btn1, btn2, btn3, btn4, btn5)

bot.send\_message(message.chat.id,

text="Привет, {0.first\_name} 👋!\nЯ новостной телеграм бот - 'Cyb3rNews'.\nЯ могу сгенерировать вам пароль, рассказать о том как обезопасить себя в интернете и показать статьи из интернета на эту тему.\n ".format(

message.from\_user), reply\_markup=markup)

def random\_password():

alphabet = string.ascii\_letters + string.digits

while True:

password = ''.join(choice(alphabet) for i in range(15))

if (any(symbol.islower() for symbol in password)

and any(symbol.isupper() for symbol in password)

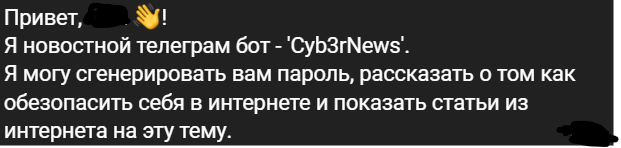
and sum(symbol.isdigit() for symbol in password) >= 1):

return password

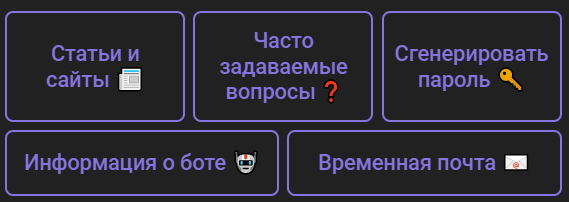
repeat = types.InlineKeyboardMarkup()

repeatbtn = types.InlineKeyboardButton(text="Сгенерировать ещё раз", callback\_data='repeatbtn')

repeat.add(repeatbtn)



*Рис. 3.3 Сообщение о начале работы с ботом*



*Рис. 3.4 Кнопки при начале работы с ботом*

Обработчик кнопок:

@bot.callback\_query\_handler(func=lambda call:True)

def callback\_query(call):

req = call.data.split('\_')

#Обработка кнопки - скрыть

if req[0] == 'unseen':

bot.delete\_message(call.message.chat.id, call.message.message\_id)

#Обработка кнопок - вперед и назад

elif 'pagination' in req[0]:

#Расспарсим полученный JSON

json\_string = json.loads(req[0])

count = json\_string['CountPage']

page = json\_string['NumberPage']

#Пересоздаем markup

markup = types.InlineKeyboardMarkup()

markup.add(types.InlineKeyboardButton(text='Скрыть', callback\_data='unseen'))

#markup для первой страницы

if page == 1:

markup.add(types.InlineKeyboardButton(text=f'{page}/{count}', callback\_data=f' '),

types.InlineKeyboardButton(text=f'Вперёд --->',

callback\_data="{\"method\":\"pagination\",\"NumberPage\":" + str(

page + 1) + ",\"CountPage\":" + str(count) + "}"))

#markup для второй страницы

elif page == count:

markup.add(types.InlineKeyboardButton(text=f'<--- Назад',

callback\_data="{\"method\":\"pagination\",\"NumberPage\":" + str(

page - 1) + ",\"CountPage\":" + str(count) + "}"),

types.InlineKeyboardButton(text=f'{page}/{count}', callback\_data=f' '))

#markup для остальных страниц

else:

markup.add(types.InlineKeyboardButton(text=f'<--- Назад', callback\_data="{\"method\":\"pagination\",\"NumberPage\":" + str(page-1) + ",\"CountPage\":" + str(count) + "}"),

types.InlineKeyboardButton(text=f'{page}/{count}', callback\_data=f' '),

types.InlineKeyboardButton(text=f'Вперёд --->', callback\_data="{\"method\":\"pagination\",\"NumberPage\":" + str(page+1) + ",\"CountPage\":" + str(count) + "}"))

#bot.edit\_message\_text(f'Страница {page} из {count}',

reply\_markup = markup, chat\_id=call.message.chat.id, message\_id=call.message.message\_id)

bot.edit\_message\_media(media=types.InputMediaPhoto

(photos[page - 1], caption=captions\_text[page - 1]), reply\_markup = markup, chat\_id=call.message.chat.id, message\_id=call.message.message\_id)

if call.data == 'repeatbtn':

bot.edit\_message\_text(chat\_id=call.message.chat.id, message\_id=call.message.message\_id,

text="Ваши сгенерированные пароли🤖:\n"

'1\.\ `' + random\_password() + '`\n' +

'2\.\ `' + random\_password() + '`\n' +

'3\.\ `' + random\_password() + '`\n' +

'4\.\ `' + random\_password() + '`\n' +

'5\.\ `' + random\_password() + '`',

parse\_mode="MarkdownV2",

reply\_markup=repeat)

@bot.callback\_query\_handler(func=lambda call:True)

def callback\_query(call):

req = call.data.split('\_')

if req[0] == 'unseen':

bot.delete\_message(call.message.chat.id, call.message.message\_id)

@bot.message\_handler(content\_types=['text'])

def func(message):

if (message.text == "Статьи и сайты 📰"):

markup = types.ReplyKeyboardMarkup(resize\_keyboard=True)

btn1 = types.KeyboardButton("Статьи 📰")

btn2 = types.KeyboardButton("Сайты на эту тему 🌐")

back = types.KeyboardButton("Назад ⬅️")

markup.add(btn1, btn2, back)

bot.send\_message(message.chat.id, text="Выберите что хотите посмотреть 🤖", reply\_markup=markup)

elif (message.text == "Статьи 📰"):

#вывод

count = 6

page = 1

markup = types.InlineKeyboardMarkup()

markup.add(types.InlineKeyboardButton(text='Скрыть', callback\_data='unseen'))

markup.add(types.InlineKeyboardButton(text=f'{page}/{count}', callback\_data=f' '),

types.InlineKeyboardButton(text=f'Вперёд --->',

callback\_data="{\"method\":\"pagination\",\"NumberPage\":" + str(

page + 1) + ",\"CountPage\":" + str(count) + "}"))

bot.send\_photo(message.from\_user.id, photos[page - 1], caption=captions\_text[page - 1], reply\_markup=markup)

elif (message.text == "Сайты на эту тему 🌐"):

inlinebtn = types.InlineKeyboardMarkup()

urlbtn1 = types.InlineKeyboardButton(text="TechRepublic",url="http://www.techrepublic.com/")

urlbtn2 = types.InlineKeyboardButton(text="Infosecurity",url="https://www.infosecurity-magazine.com/")

urlbtn3 = types.InlineKeyboardButton(text="Zero Day from ZDNet",url="http://www.zdnet.com/topic/security/")

urlbtn4 = types.InlineKeyboardButton(text="CIO", url="http://www.cio.com/")

urlbtn5 = types.InlineKeyboardButton(text="The Security Ledger", url="https://securityledger.com/")

inlinebtn.add(urlbtn1, urlbtn2 , urlbtn3, urlbtn4, urlbtn5)

bot.send\_message(message.chat.id, text="Cайты на тему цифровой гигиены 🦠 :", reply\_markup=inlinebtn)

elif (message.text == "Часто задаваемые вопросы ❓"):

markup = types.ReplyKeyboardMarkup(resize\_keyboard=True)

btn1 = types.KeyboardButton("Цифровая гигиена❓")

btn2 = types.KeyboardButton("Советы по цифровой гигене 🦠")

btn3 = types.KeyboardButton("Безопасный пароль 🔑")

btn4 = types.KeyboardButton("Что такое временная почта 📧")

back = types.KeyboardButton("Назад ⬅️")

markup.add(btn1, btn2, btn3 , btn4, back)

bot.send\_message(message.chat.id, text="Выберите вопрос 🤖", reply\_markup=markup)

elif (message.text == "Цифровая гигиена❓"):

bot.send\_message(message.chat.id, "Цифровая гигиена – это свод правил, следуя которым, человек обеспечивает себе информационную безопасность (не анонимность, а защиту) в сети Интернет. Относится к сфере знаний о цифровой безопасности.")

elif (message.text == "Что такое временная почта 📧"):

bot.send\_message(message.chat.id, "Временная почта — ящик для разового использования. Он создается в специальных сервисах в интернете на короткое время (например, 10 минут).В остальном это такой же почтовый клиент по типу Gmail: на него можно получать почту и читать письма.")

elif message.text == "Советы по цифровой гигене 🦠":

bot.send\_message(message.chat.id, text="7 правил цифровой гигиены, за которые вы скажете себе спасибо:\n1. Устанавливайте сложные пароли и регулярно меняйте их\n2. Делайте резервное копирование данных\n3. Не делитесь слишком личной информацией в соцсетях\n4. Регулярно проверяйте историю финансовых операций\n5. Периодически отписывайтесь от лишнего\n6. Следите за тем, какой цифровой отпечаток вы оставляете\n7. Регулярно обновляйте программное обеспечение")

elif message.text == "Безопасный пароль 🔑":

bot.send\_message(message.chat.id, text="Безопасный пароль содержит комбинирование шести или больше строчных и прописных букв, плюс знаки препинания и цифры.\nИспользуйте все четыре типа написания – это надёжнее всего.\nНапример, замените «добро пожаловать» на W3Lc0mE ^ 9. 8 и более символов должно содержаться в пароле.")

elif message.text == "Информация о боте 🤖":

link = types.InlineKeyboardMarkup()

linkbtn = types.InlineKeyboardButton(text="Telegram", url="https://t.me/eternal\_awake")

linkbtn1 = types.InlineKeyboardButton(text="VK", url="https://vk.com/lovelynner")

link.add(linkbtn, linkbtn1)

bot.send\_message(message.chat.id, text="Telegram-бот был создан одним разработчиком для людей которые не знают азы цифровой гигиены.\nДля сотрудничества писать:", reply\_markup=link)

elif message.text == "Сгенерировать пароль 🔑":

bot.send\_message(message.from\_user.id,text="Ваши сгенерированные пароли🤖:\n"

'1\.\ `' + random\_password() + '`\n' +

'2\.\ `' + random\_password() + '`\n' +

'3\.\ `' + random\_password() + '`\n' +

'4\.\ `' + random\_password() + '`\n' +

'5\.\ `' + random\_password() + '`', parse\_mode="MarkdownV2", reply\_markup=repeat)

elif message.text == "Временная почта 📧":

url = types.InlineKeyboardMarkup()

url1 = types.InlineKeyboardButton(text="Ссылка на сайт", url="https://temp-mail.org/")

url.add(url1)

bot.send\_message(message.chat.id, text="TempMail📧 — это одноразовый почтовый ящик, который хранит письма на короткий срок. Сервис работает по принципу временной почты (или почты на 10 минут).Это удобно в том случае когда вам нужно зарегистрироваться на сайте и не получать от него spam сообщения на почту.", reply\_markup=url)

elif (message.text == "Назад ⬅️"):

markup = types.ReplyKeyboardMarkup(resize\_keyboard=True)

btn1 = types.KeyboardButton("Статьи и сайты 📰")

btn2 = types.KeyboardButton("Часто задаваемые вопросы ❓")

btn3 = types.KeyboardButton("Сгенерировать пароль 🔑")

btn4 = types.KeyboardButton("Информация о боте 🤖")

btn5 = types.KeyboardButton("Временная почта 📧")

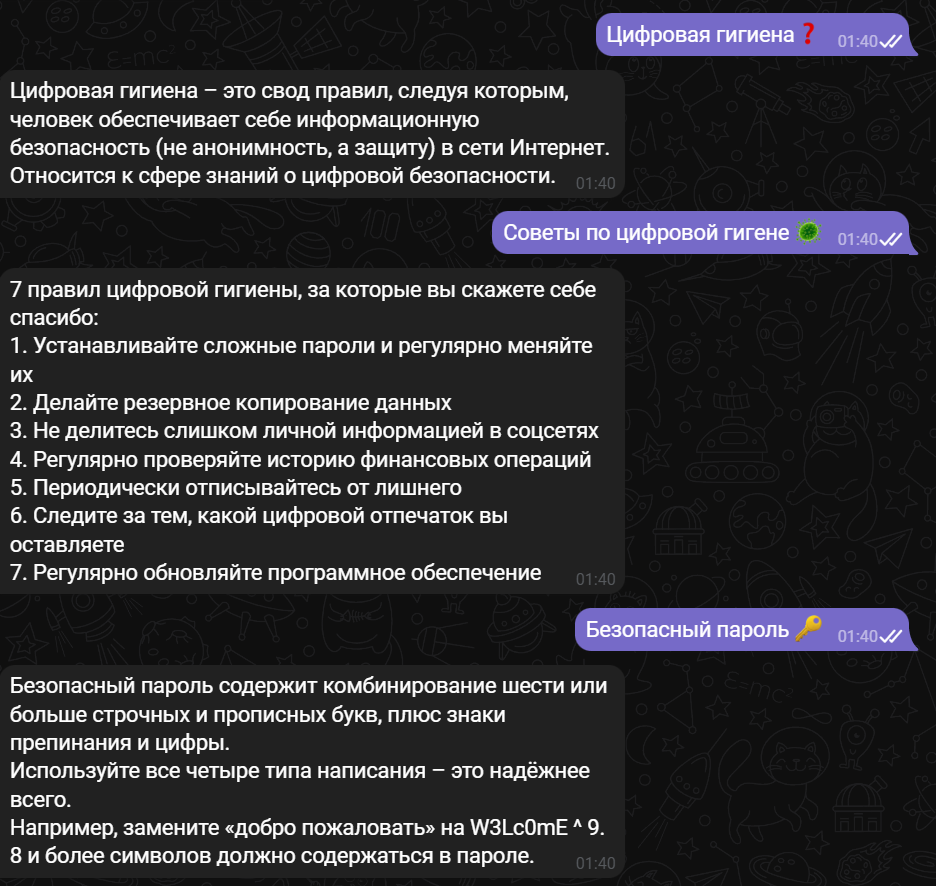
markup.add(btn1, btn2, btn3 , btn4 , btn5)

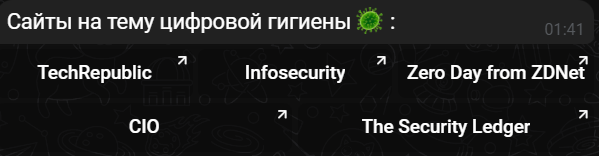
bot.send\_message(message.chat.id, text="Вы вернулись в главное меню 🤖", reply\_markup=markup)

else:

bot.send\_message(message.chat.id, text="На такую комманду я не запрограммировал 🤖")

bot.polling(none\_stop=True)

****

*Рис 3.5 Частые вопросы*****

*Рис 3.6 Сообщение с кнопками на полезные сайты*

# **4. ТЕСТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ**

В данной главе будет отображена история изменений программного обеспечения, проверена версионность разработанного приложения, определены объекты тестирования, а также описание процесса тестирования.

## **4.1 История изменений**

Версии разработанного программного продукта можно отследить в таблице (табл. 4.1). На данной таблице будут указаны изменения частей программного обеспечения, а также даты выхода изменений.

Таблица 4.1

Версии программного продукта

|  |  |  |
| --- | --- | --- |
| Версия | Дата выхода | Изменения |
| 1 | 2 | 3 |
| 1.0 | 16.12.22 | Создано:   * пустая оболочка бота; * регистрация бота в BotFather; |
| 1.1 | 17.12.2022 | Добавлено:   * команды для работы с ботом; * кнопки быстрого доступа; * несколько страниц для примера отображения; |
| 1.2 | 18.12.2022 | Добавлено:   * генератор паролей; |
| 2.0 | 19.12.2022 | Изменено:   * общий вид бота; * иконка бота; * функционал кнопок; |
| 2.1 | 20.12.2022 | Добавлено:   * нажимаемые кнопки на сообщениях бота; |
| 2.2 | 20.12.2022 | Добавлено:   * поддержка редактирования сообщений ботом; |

## **4.2 Определение объектов тестирования**

Существует две группы объектов тестирования – функциональные и нефункциональные.

К функциональному тестированию относится:

* тестирование создания паролей;
* тестирование команд;
* тестирование кнопок;
* тестирование кнопок на сообщениях;
* тестирование ссылок;

К нефункциональному тестированию относится:

* тестирование пользовательского интерфейса.

## **4.3 Описание процесса тестирования**

Описание процесса функционального тестирования приведено в таблице (табл. 4.2).

Таблица 4.2

Функциональное тестирование. Тест-кейсы

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| №  п/п | Название | Тестовые данные | Ожидаемый результат | Фактический результат | Статус |
| 1. | Создание паролей | Кнопка «Сгенерировать пароли» | Генерация несколько хорошо защищенных паролей | Создалось 5 паролей | Пройден |
| 2. | Приветствие бота при команде /start | /start | Сообщение о приветствии с указанием ника пользователя | Сообщение отобразилось, ник указан верно | Пройден |
| 3. | Работа ссылок | Ссылка на Telegram, ВКонтакте | Ссылки успешно откроются, пересылая на верную страницу | Телеграм ссылка открыла чат с создателем бота, ссылка ВК переслала на верный аккаунт | Пройден |
| 4. | Листание постов | Кнопки на постах  <- и -> | Произойдет смена на следующий пост | Пост сменился, само сообщение при этом редактировалось, а не удалялось | Пройден |

Нефункциональное тестирование:

Было проведено тестирование графического интерфейса пользователя.

Общая проверка:

* орфографические и пунктуационные ошибки отсутствуют;
* все окна имеют правильные заголовки и названия;

Посты:

* картинки присутствуют;
* цвет шрифта соответствует требованиям;
* размер шрифта соответствует требованиям;

Кнопки:

* цвет кнопок меняется в зависимости от темы пользователя;
* цвет шрифта кнопок соответствует требованиям;
* при наведении курсора на кнопку, кнопка меняет цвет;

В результате были протестированы основной функционал приложения и графический интерфейс пользователя.

# **5. МЕРОПРИЯТИЯ ПО ТЕХНИКЕ БЕЗОПАСНОСТИ И ОХРАНЕ ТРУДА.**

Техника безопасности при работе с компьютером на предприятии предусматривает наличие общедоступной инструкции, в которой указаны обязательные требования к обустройству рабочего места и процессу использования техники. Эти правила едины для всех организаций, их выполнение контролируется руководящими органами.

Основными нормативными документами, которые содержат обязательные требования, можно считать:

Трудовой кодекс;

санитарно-гигиенические нормы и правила СанПиН 2.2.2/2.4.1340-03, утвержденные постановлением Главного государственного санврача от 13 июня 2003 года N 118;

межгосударственный стандарт ГОСТ 12.0.003-2015, который содержит список основных вредных и опасных производственные факторов и их классифицирующие признаки;

типовая инструкция ТОИ Р-45-084-01, в которой содержатся основные правила для работников, использующих компьютер;

другие нормативные документы.

Требования перечисленных правовых актов, в которых содержатся общие правила защиты персонала при выполнении трудовых обязанностей и техника безопасности во время работы за компьютером, обязательны к исполнению всеми работниками этой категории.

На специалиста, работающего за компьютером, могут воздействовать опасные и вредные факторы, такие как:

высокая температура, характерная для отдельных элементов компьютерной техники и создающая общий повышенный температурный фон в рабочем помещении;

высокая степень монотонности рабочего процесса;

значительный уровень зрительный нагрузок, испытываемых работником;

риск поражения статическим электричеством при случайном контакте с отдельными элементами техники;

недостаточная освещенность рабочей зоны;

повышенный уровень напряженности электрического и магнитного полей, генерируемых работающей техникой;

высокий уровень блёскости и контрастности рабочего экрана, негативно влияющий на зрение работника в длительной перспективе;

повышенная яркость светового изображения;

повышенный уровень пульсации светового потока;

повышенное значение напряжения в электрической цепи, замыкание которой может произойти через тело человека;

другие факторы.

Чтобы донести до работника требования, которые устанавливает техника безопасности при использовании компьютера, работодатель обязан организовать инструктаж. Он проводится на основании специально составленной инструкции, которую обычно разрабатывает специалист по охране труда или непосредственный руководитель работника. Проведение инструктажа нужно организовать до начала самостоятельной работы: специалист допускается к работе только после его прохождения. Это значит, до него донесены правила техники безопасности при работе с компьютером. Инструкция- памятка по технике безопасности за компьютером должна быть выдана каждому работнику в бумажном или электронном виде.

Техника безопасности в начале работы на компьютере требует выполнения следующих операций:

проверить исправность элементов электросистемы, обеспечивающей питание компьютера, включая электропроводку, выключатели, вилки и розетки, при помощи которых аппаратура подключается к сети;

проконтролировать заземление компьютера;

проверить его работоспособность.

При выполнении работ нужно соблюдать следующие правила:

запрещается класть на корпус и дисплей компьютера посторонние предметы, прикасаться к элементам аппаратуры мокрыми руками, производить чистку корпуса оборудования, находящегося под напряжением, располагать технику близко к жилищно-коммунальным инженерным системам;

в случае обнаружения неисправности компьютера немедленно прекратить работу и сообщить об этом непосредственному руководителю;

эксплуатировать компьютер только с соблюдением инструкции, установленной производителем;

избегать частого и необоснованного включения и выключения компьютера во время работы.

После завершения работы сотруднику нужно выполнить следующие действия:

выключить компьютер с использованием алгоритма, установленного производителем;

обесточить периферийное оборудование;

убедиться в отключении техники;

выполнить очистку рабочих поверхностей влажной тканью.

Требования к расположению работника за компьютером нацелены на обеспечение его комфорта в течение всей рабочей смены и отсутствие негативных следствий длительной работы. Они действуют для любых рабочих мест, будь то бухгалтерия, обучающий класс или кабинет информатики. Они включают следующие правила:

полная опора ступнями на пол при посадке;

использование компьютерной мебели, отвечающей нормам СанПиН 2.2.2/2.4.1340-03;

отказ от скрещивания конечностей, способного затруднить кровообращение;

соблюдение расстояние до монитора компьютера не меньше 45 сантиметров;

правильная установка освещения, которое не должно светить в глаза и оставлять блики на рабочем мониторе;

клавиатура размещается за 20-30 сантиметров от края стола;

стул стоит таким образом, чтобы спина лишь немного упиралась в его спинку;

локти согнуты под прямым углом, а в кистях рук, лежащих на столе, не чувствуется напряжения;

локти не висят в воздухе, а комфортно располагаются на подлокотниках кресла или столешнице. Их позиция существенно не меняется при передвижении мышки;

ноги упираются в твердую поверхность, распрямлены вперед, а не подогнуты под себя;

если пользователь носит очки, то нужно убедиться в том, что он может свободно регулировать угол наклона экрана.

Для облегчения длительной работы рекомендуется использовать вспомогательное оборудование, например, особые очки, антибликовые мониторы, коврики для мыши и проч. Нарушения техники безопасности при работе на компьютере способны вызвать стойкие расстройства здоровья, которые потом будет трудно ликвидировать.

Длительность непрерывного труда и отдыха для сотрудников, постоянно работающих за персональной офисной техникой, регламентирована на законодательном уровне. Соответствующие нормы приводятся в СанПиН 2.2.2/2.4.1340-03, который устанавливает обязательную продолжительность перерывов в такой работе. Они зависят от длины рабочей смены сотрудника и определяются так:

при 8-часовой рабочей смене в зависимости от тяжести и напряженности труда работника общая продолжительность перерывов за рабочий день должна составлять от 50 до 90 минут;

при 12-часовой рабочей смене – от 80 до 140 минут.

Проводить такие перерывы за монитором, тратя их на чтение новостей или онлайн- игры, нельзя. Работнику нужно посвятить их физическим упражнениям, короткой прогулке, гимнастике для глаз и другой активности.

# **ЗАКЛЮЧЕНИЕ**

В результате выполнения курсового проекта на тему «Разработка сервиса для оценки эмоционального выгорания ИТ специалистов» было разработано мобильное приложение, позволяющий в удобном формате отследить и предотвратить состояние эмоционального выгорания сотрудников.

В процессе моделирования программного продукта были построены следующие диаграммы:

* диаграмма вариантов использования для роли гостя;
* диаграмма вариантов использования для роли пользователя;
* диаграмма вариантов использования для роли ИИ;
* диаграмма последовательностей «Отметка своего состояния»;
* диаграмма последовательностей «Регистрация»;
* диаграмма последовательностей «Подтверждение email»;
* диаграмма активности «Тестирование»;
* диаграмма активности «Рекомендации на основе тестирования»;
* диаграмма активности «Регистрация»;
* диаграмма активности «Авторизация».

В процессе выполнения курсовой работы была разработана база данных SQL, содержащая таблицы «Вопросы», «Пользователи», «Состояние», «Записи в дневнике», «Результаты».

Итоговый разработанный web-сайт выполняет следующие функции:

* авторизация пользователей;
* создание заказов;
* удаление заказов;
* подтверждение роли пользователя;
* просмотр списка созданных заказов;
* добавление оборудования;
* удаление оборудования.

Проведенное тестирование показало, что функционал программного продукта работает корректно, в соответствии с установленными требованиями.

Графический интерфейс интуитивно понятен, не требует длительного времени на освоение, работает корректно.

# **СПИСОК ЛИТЕРАТУРЫ**

1. Маркин, А.В. Построение запросов и программирование на SQL [Текст]: Учебное пособие / А.В. Маркин - Рязань: РГРТУ, 2008. - 312с. - ISBN 978-5-7722-0285-2.
2. Клайв К. SQL [Текст]: Справочник / К. Клайв, Д. Клайв, Б. Хант - М.: "Кудиц-Образ", 2006. - 832с. - ISBN 5-9579-0114-8, ISBN 5-596-00481-8.
3. Дейт К., Руководство по реляционной СУБД DB2 [Текст] / Когаловский М.Р. - М.: "Финансы и статистика", 1988 - 320c. - ISBN 5-279-0063-9 (в пер.).
4. И. Аделекан Kotlin: программирование на примерах. 1-е издание, БХВ-Петербург, 2020.
5. Ф. Стюарт Android. Программирование для профессионалов. 4-е издание; ООО Издательство "Питер", 2020
6. Основы UML – диаграммы использования (use-case).- URL: https://pro-prof.com/archives/2594
7. Фаулер М., Скотт К UML. Основы СПб.: Символ, 2006, 184 с.
8. Ван, Тассел Д. Стиль, разработка, эффективность, отладка и испытания программ / Ван Тассел Д.. - М.: Мир, 2017.
9. Вирт, Н. Алгоритмы и структуры данных / Н. Вирт. - М.: Мир, 2016.