**ОПИСАНИЕ ФУНКЦИОНАЛЬНЫХ ХАРАКТЕРИСТИК ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ АВТОМАТИЗАЦИИ ВЗАИМОДЕЙСТВИЯ С ИНТЕЛЛЕКТУАЛЬНОЙ СИСТЕМОЙ**

г. Москва

2024

**ОГЛАВЛЕНИЕ**

[1 ОБЩИЕ СВЕДЕНИЯ 3](#_Toc178325253)

[1.1 Цели создания системы 3](#_Toc178325254)

[1.2 Назначение системы 3](#_Toc178325255)

[2 АРХИТЕКТУРА ПРОЕКТА 3](#_Toc178325256)

[3 ПЕРЕЧЕНЬ ПОДСИСТЕМ, ИХ НАЗНАЧЕНИЕ И ОСНОВНЫЕ ХАРАКТЕРИСТИКИ 5](#_Toc178325257)

[3.1 Подсистема авторизации пользователей 5](#_Toc178325258)

[3.2 Подсистема навигации пользователя в боковом меню 5](#_Toc178325259)

[3.3 Подсистема чатов 5](#_Toc178325260)

[3.4 Требования к контейнеризации](#_Toc178325262) 5

[3.5 Требования к программному обеспечению 7](#_Toc178325263)

[4 ТРЕБОВАНИЯ К НАДЕЖНОСТИ И БЕЗОПАСНОСТИ ТЕХНИЧЕСКИХ СРЕДСТВ И ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ 10](#_Toc178325264)

# ОБЩИЕ СВЕДЕНИЯ

Полное наименование: Программное обеспечение автоматизации взаимодействия с интеллектуальной системой (далее Система).

## Цели создания системы

Система создана с целью автоматизации процессов выполнения различных задач, таких как генерация текстов, ответы на вопросы, обработка информации и помощь в решении проблем

## Назначение системы

Система предназначена для использования в разных сценариях, начиная от повседневных запросов и заканчивая сложными бизнес-задачами. Мобильное приложение позволяет принимать от пользователя запросы в виде текстовых сообщений и генерировать ответ

# АРХИТЕКТУРА ПРОЕКТА

Система разработана в рамках микросервисной архитектуры с использованием современных технологий и фреймворков для обеспечения эффективности, масштабируемости и удобства в поддержке. Архитектура состоит из следующих ключевых компонентов:

1. Backend (FastAPI):
* Фреймворк FastAPI используется для разработки серверной части приложения, обеспечивает удобное создание API для взаимодействия с Frontend-частью Системы.
1. База данных (PostgreSQL):
* PostgreSQL выбрана в качестве системы управления базами данных для обеспечения высокой надежности и поддержки сложных запросов.
* SQL Alchemy ORM используется для удобного взаимодействия с базой данных, создания моделей данных и выполнения запросов.
* Внедрение миграций Liquibase обеспечивает контроль версий схемы базы данных и упрощает процесс обновления структуры данных.
1. Mobile Frontend (Flutter):
* Frontend разработан с использованием фреймворка Flutter для создания динамичного пользовательского интерфейса.
* Используются компоненты Material UI для создания модульной структуры приложения.
* Для управления состоянием приложения используется Bloc.
* Маршрутизация реализована с использованием GoRouter для навигации по экранам.
* Интерактивные элементы интерфейса обеспечивают приятный пользовательский опыт.
1. Контейнеризация:
* Применение Docker-контейнеров для сервисов Системы обеспечивает легкость развертывания, масштабирования и управления зависимостями приложения.
* Openshift используется для управления многоконтейнерными приложениями.

Эта архитектура предоставляет гибкую и масштабируемую основу для развертывания, обеспечивая эффективное взаимодействие между всеми сервисами проекта.

# ПЕРЕЧЕНЬ ПОДСИСТЕМ, ИХ НАЗНАЧЕНИЕ И ОСНОВНЫЕ ХАРАКТЕРИСТИКИ

## Подсистема авторизации пользователей

Подсистема авторизации пользователей представляет собой компонент Системы, который отвечает за управление доступом пользователей к различным ресурсам и функциям. Она обеспечивает проверку тарифа пользователя на выполнение определённых действий и доступ к количеству запросов. Основные функции подсистемы авторизации включают:

1. Аутентификация;
2. Авторизация;
3. Контроль и управление сессиями;
4. Логирование и мониторинг.

## Подсистема навигации пользователя в боковом меню

Подсистема навигации пользователя в боковом меню позволяет пользователю перемещаться по разделам Системы и представляет собой набор данных о:

1. чате изображений;
2. списку текстовых чатов;
3. информации о тарифе.

## Подсистема чатов

После успешной авторизации пользователю в боковом меню необходимо выбрать необходимый чат. На выбор дается один чат изображений для генерации изображений по описанию и множество текстовых чатов с поддержкой контекста.

**ТРЕБОВАНИЯ К СЕРВЕРНОЙ ИНФРАСТРУКТУРЕ**

## Требования к контейнеризации

Контейнеризация приложений в OpenShift требует соблюдения ряда принципов и стандартов, чтобы гарантировать эффективную работу и управление контейнерами. Вот основные требования:

1. Изоляция приложений: Контейнеры должны быть изолированы друг от друга и от хост-системы.

2. Использование образов контейнеров: Приложения должны быть упакованы в образы контейнеров, которые могут быть созданы с использованием Dockerfile или других инструментов сборки. Эти образы должны быть доступны в реестре контейнеров.

3. Поддержка многопоточности и масштабируемости: Приложения должны быть спроектированы для работы в распределенной среде, что включает возможность горизонтального масштабирования.

4. Сетевые настройки: Приложения должны использовать сетевые политики OpenShift для управления сетевым трафиком и обеспечения безопасности.

5. Хранение данных: для хранения данных необходимо использовать постоянные тома (Persistent Volumes) и соответствующие классы хранения (Storage Classes).

6. Мониторинг и логирование: Приложения должны быть настроены для интеграции с системами мониторинга и логирования, такими как Prometheus.

7. Безопасность: следует учитывать безопасность на всех уровнях, включая использование Security Contexts, ограничение прав доступа и применение политик безопасности.

8. Конфигурация и секреты: Конфигурационные данные и секреты должны храниться в ConfigMaps и Secrets соответственно, чтобы обеспечить безопасное управление конфиденциальной информацией.

9. CI/CD интеграция: рекомендуется интегрировать приложения с CI/CD пайплайнами для автоматизации сборки, тестирования и развертывания.

10. Соблюдение стандартов и практик DevOps: важно следовать лучшим практикам DevOps при разработке, развертывании и управлении приложениями.

Соблюдение этих требований поможет обеспечить стабильную и безопасную работу приложений в среде OpenShift.

## Требования к программному обеспечению

Функциональные требования описывают конкретные функции и возможности, которые должно обеспечивать приложение DLChat.

1. Обработка пользовательских запросов: приложение должно уметь принимать текстовые запросы от пользователей.

* Возможность ввода текста в текстовое поле.
* Ограничение длины вводимого текста (например, не более 2048 символов за один запрос).
* Поддержка специальных символов, пробелов и многострочного ввода.
* Поддержка мультиязычного ввода.

2. Генерация текстового ответа: приложение должно генерировать текстовые ответы на основе введенного пользователем запроса с помощью модели GPT.

* Генерация ответов на естественном языке в соответствии с введенным запросом.
* Ответ должен быть последовательным, соответствовать контексту и сохранять логику диалога.
* Ответы должны быть синтаксически правильными и, насколько возможно, грамматически корректными.
* В случае невозможности дать точный ответ, приложение должно сообщать об этом корректно, без предоставления неверной информации.

3. Поддержка диалогов: система должна уметь поддерживать многошаговые диалоги с пользователем, сохраняя контекст предыдущих сообщений.

* Возможность ответов, основанных на предыдущих сообщениях в рамках одного диалога.
* Хранение контекста общения до завершения текущей сессии.
* Умение восстанавливать логику беседы и корректно отвечать на последующие вопросы пользователя в пределах одной сессии.

4. Обработка ошибок и нераспознанных запросов: приложение должно корректно обрабатывать ошибки пользователя или случаи, когда запрос не может быть выполнен.

* Генерация сообщений об ошибках или информационных сообщений в случае, если система не может предоставить ответ (например, при непонятном запросе или при запросах за пределами возможностей модели).
* Приложение должно предложить пользователю переформулировать запрос или задать уточняющий вопрос.

5. Интернационализация и поддержка нескольких языков: приложение должно поддерживать взаимодействие на нескольких языках.

* Обработка запросов на разных языках, включая (но не ограничиваясь): английский, испанский, русский, китайский, немецкий, французский и другие.
* Генерация ответов на языке исходного запроса.
* Возможность определения языка ввода автоматически без необходимости изменения настроек.

6. Генерация текстов различного назначения: приложение должно предоставлять пользователю возможность генерировать тексты для различных целей.

* Создание текстов по запросам пользователя, например, статьи, письма, стихи, резюме, эссе, код программирования и т. д.
* Поддержка генерации кода на различных языках программирования (например, Python, JavaScript, C++).
* В случае генерации текста для конкретных задач (например, коммерческие письма), приложение должно учитывать тональность, форматирование и стиль текста в соответствии с указаниями пользователя.

7. Функция редактирования и улучшения текста: приложение должны иметь возможность загружать текст для редактирования или улучшения.

* Возможность загрузки текста в текстовое поле для проверки.
* Автоматическое исправление грамматических, синтаксических и стилистических ошибок.
* Рекомендации по улучшению текста на основе тональности и структуры.

8. Перевод текста: приложение должно предоставлять возможность перевода текстов между языками.

* Возможность ввода текста на одном языке и генерации перевода на другой язык.
* Поддержка основных языковых пар (например, английский ↔ русский, испанский ↔ немецкий).
* Корректное отображение переведенного текста с сохранением исходного смысла и структуры.

9. Анализ и резюмирование текста: система должна уметь анализировать и резюмировать длинные тексты.

* Поддержка загрузки текста для анализа (например, отчеты, статьи, документы).
* Генерация краткого резюме на основе анализа текста.
* Возможность выделения ключевых идей и основных моментов из длинного текста.

10. Поддержка интерактивного обучения: система должна предоставлять возможность обучения пользователя через объяснение сложных тем или предоставление примеров.

* Возможность объяснения технических или научных понятий простыми словами.
* Ответы должны содержать примеры или иллюстрации для лучшего понимания (например, примеры кода, формулы, схемы).
* Возможность пошагового объяснения для тем, требующих последовательного подхода.

11. Персонализация и настройка стиля ответов: пользователь должен иметь возможность настроить стиль ответа DLChat под свои нужды.

* Возможность выбора тональности (официальный, дружелюбный, нейтральный, технический и т.д.).
* Возможность настройки длины ответа (краткий, развернутый).
* Возможность переключения между стилями объяснения (например, "объясни как ребенку", "объясни технически").

12. Нефункциональные требования

12.1. Производительность

- Скорость обработки: быстрая регистрация и обработка запроса пользователя.

- Масштабируемость: способность системы обрабатывать большое количество запросов в периоды пиковых нагрузок.

12.2. Надежность и отказоустойчивость

- Резервное копирование: регулярное резервное копирование данных.

- Восстановление после сбоев: способность системы быстро восстанавливаться после сбоев.

12.3. Безопасность

- Защита данных: защита данных от несанкционированного доступа и утечки.

- Аутентификация и авторизация: надежные механизмы аутентификации и авторизации пользователей.

12.4. Поддержка и обслуживание

- Техническая поддержка: круглосуточная поддержка пользователей системы.

- Обновления и улучшения: регулярные обновления ПО для улучшения функциональности и безопасности.

# ТРЕБОВАНИЯ К НАДЕЖНОСТИ И БЕЗОПАСНОСТИ ТЕХНИЧЕСКИХ СРЕДСТВ И ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Требования к надежности технических средств и программного обеспечения включают в себя несколько аспектов, целью которых является обеспечение стабильности и бесперебойной работы системы, а именно:

1. Доступность: системы должны быть доступны пользователям в любое время, согласно установленным уровням обслуживания (SLA). Это включает в себя минимизацию времени простоя.

2. Отказоустойчивость: системы должны быть спроектированы так, чтобы продолжать функционировать даже в случае сбоев отдельных компонентов. Это может включать резервирование, дублирование и автоматическое переключение на резервные системы.

3. Устойчивость к нагрузке: программное обеспечение должно быть способно обрабатывать ожидаемую и даже повышенную нагрузку без снижения производительности.

4. Поддержка и обслуживание: надежные системы должны иметь возможность для легкого обновления, исправления ошибок и масштабирования без значительных перебоев в работе.

5. Тестирование и валидация: тестирование (включая нагрузочные и стресс-тесты) должно проводиться для выявления потенциальных проблем до их появления в рабочей среде.

Требования к безопасности включают в себя следующие основные пункты:

1. Аутентификация и авторизация: необходимо обеспечить надежные механизмы аутентификации пользователей и контроля доступа к системам и данным.

2. Шифрование данных: данные должны быть защищены как в состоянии покоя, так и при передаче. Использование протоколов шифрования (например, TLS) является обязательным.

3. Защита от угроз: системы должны иметь средства защиты от различных угроз, таких как вирусы, вредоносное ПО, атаки DDoS и другие виды кибератак.

4. Мониторинг и аудит: необходимо внедрить системы мониторинга для отслеживания подозрительной активности и ведения журналов для последующего анализа.

5. Обновления и патчи: регулярное применение обновлений безопасности и исправлений для программного обеспечения и операционных систем.

6. Обучение пользователей: пользователи должны быть обучены основам безопасности, включая распознавание фишинга и безопасное использование систем.

7. Политики безопасности: разработка и внедрение документированных политик безопасности, которые определяют правила и процедуры для защиты информации.

8. Резервное копирование данных: регулярное создание резервных копий критически важных данных для обеспечения их восстановления в случае потери или повреждения.

Соблюдение этих требований поможет обеспечить как надежность, так и безопасность технических средств и программного обеспечения, что, в свою очередь, повысит доверие пользователей и снизит риски для бизнеса.