Оглавление

1 Введение 2
2 Итоги 1 аттестации 2
2.1 Организационная работа 2
2.2 Составление предпроектного исследования и технического задания 3
2.2.1 Предпроектное исследование 3
2.2.2 Техническое задание 4
2.3 Проблемы 5
2.4 Итоги этапа 6
3 Разработка и защита MVP 6
3.1 Разработка Back-end 8
3.1.1 Модуль «Авторизация/аутентификация» 8
3.1.2 Модуль «Подписки на сервис» 8
3.1.3 Модуль «Компания» 8
3.1.4 Модуль «AI ассистента» 8
3.1.5 Модуль «Автоответчик» 9
3.1.6 Модуль «Интеграции с внешними API» 9
3.1.7 Модуль «Диалоги» 10
3.1.8 Связь модулей и взаимодействие 10
3.2 Разработка Front-end 10
4 Критерии успешности 2 этапа. 11

1 Введение

Проект «Автоответчик для почты и других чатов» (DialogX) представляет собой веб-сайт приложение для автоматизации коммуникации. По состоянию на конец марта 2025 года завершен этап первой аттестации. В данном отчете представлены:

- результаты выполнения работ на первом этапе;
- анализ возникших сложностей и нахождение их решений;
- план проекта на вторую аттестацию (апрель 2025);
- оценка рисков и стратегия по их минимизации;
- Выделение критериев успешности 2 аттестации.

2 Итоги 1 аттестации

2.1 Организационная работа

Началом разработки DialogX стало создание GitHub-репозитория. Репозиторий включал в себя несколько веток, что дает возможность вести параллельную разработку отдельных модулей, а также избежать хаотичных изменений и конфликтов при слиянии кода. Для коммитов было определено следующее правило, каждый коммит должен включать в себе Id и комментарий, который пояснял что именно было добавлено/изменено в данном коммите. Id для коммита брался из Id задачи в таск-трекере Jira.

Для координации команды были использованы два основных канала коммуникации. Еженедельные созвоны в Discord стали ключевым элементов коммуникации команды: каждую пятницу мы подводили итого, разбирали возникшие проблемы, обсуждали текущие задачи, распределяли задачи на следующую неделю. Эти встречи длились около часа-двух часов и включали не только обсуждение технических вопросов, но и критический анализ выполненной работы – что хорошо, а что надо переделать. Для оперативного

общения в течение дня был создан Telegram-чат, где члены команды оперативно решали текущие вопросы.

Управление задачами велось через таск-трекер Jira, где был организован борд с колонками «ТЗ после спринта», «На паузе», «Планирование», «В работе», «Тестирование», «Готово». Также у задачи были определены сроки, написано описание задачи, указывался исполнитель, а также задачи можно было объединять в эпики.

2.2 Составление предпроектного исследования и технического задания

На этапе первой аттестации основной задачей было написание предпроектного исследования и технического задания, а также составление UML диаграмм, ER диаграммы, схемы API, выбор стека технологий, составление дизайн-макетов. Мы начали с предпроектного исследования

2.2.1 Предпроектное исследование

В ходе предпроектного исследования был проведен анализ 4 конкурентов, целевой аудитории и рынка, а также была составлена финансовая модель и дорожная карта.

Анализ конкурентов включал в себя SWOT-анализ, а также бенчмаркинг конкурентов. Благодаря анализу конкурентов были сформированы уникальные преимущества DialogX, которые включают в себя: интеграцию с email, возможность добавления виджета чата на сторонний сайт, наличие AI-ответов и динамической системы ценообразования стоимости подписки.

Целевой аудиторией нашего сервиса стали малые и средние бизнесы, а так простые люди, которые осуществляют активную коммуникацию через мессенджеры. Географически продукт ориентирован на российский рынок с потенциалом выхода на рынок стран СНГ.

В качестве основной модели монетизации была выбрана подписка с тремя тарифами: тестовым (бесплатный пробный 7 дневный период), соло (для

1- го пользователя) и командный(динамическое ценообразование, которое зависит от числа пользователей). Также была просчитана базовая UNIT-экономика для разных сценариев поведения лидов. Кроме того был рассчитан P&L и ROI и составлена дорожная карта проекта.

2.2.2 Техническое задание

Результаты предпроектного исследования повлияли на формирование ТЗ и частично легли в его основу. Перечислим наиболее значимые аспекты, которые включало ТЗ:

- перечень документов, на основании которых создается приложение;
- наименование заказчика и исполнителя;
- были выделены и определены цели и назначение web-сайта;
- функциональные требования: разделение по ролям (пользователь, оператор, администратор), работа с чатами, шаблонными ответами и сделками;
- нефункциональные требования: **производительность**, а именно быстрая загрузка страниц (<2 сек) при стабильной скорости интернета (50–100 Мбит/с.), **безопасность**, а именно шифрование данных (HTTPS), аутентификация/авторизация (Spring Security), **кросс-платформенность,** а именно доступность для использования на большинстве популярных браузеров (Яндекс, Гугл, Опера) и т.д;
- архитектура: UML-диаграммы взаимодействия компонентов, ER-диаграмма БД, схема API;
- дизайн: макеты основных страниц в Figma, UI kit с компонентами интерфейса и фирменной цветовой палитрой, а также брэнд-бук.

2.3 Проблемы

На начальном команда столкнулась с проблемой переоценки своих возможностей и неправильного распределения времени задач. Часть задач не укладывались в сроки и переносились на следующие спринты. Это создавало «эффект снежного кома» - накопление невыполненных задач.

Причинами стала оптимистичная оценка сложностей задач без учета непредвиденных трудностей и отсутствие приоритизации – часть времени уходила на второстепенные задачи в ущерб необходимым.

Для решения данных проблем на еженедельных созвонах проводились митинги с переоценкой трудозатрат. А также выделялись критические задачи (с наивысшим приоритетом), их обозначали «Must-have» для спринта. Кроме того начали создаваться еженедельные отчеты, которые включают в себя состояние задач (выполненные, перенесенные на следующий спринт, проваленные), разбор проблем и перечень изменений в бэклоге.

Еще одной проблемой стало сложность с взаимопроверкой между команд. По условиям аттестации, команды должны были проверить проекты друг друга.

Однако, 3 из 4 команд загрузили работы на Github в последний день дедлайна, что создало резкую нагрузку при проверке на команду. Необходимо было проверить команды в последний день, дать обратную связь, а также при наличии правок, сделанных командой после проверки, перепроверить исправленные моменты и изменить баллы в чек листе.

Наша команда выложила проекта за 2 дня до срока, но проверяющие команды предоставили фидбек в последний день(некоторые за несколько часов до конца), выявив недочеты в нашем проекте. Пришлось экстренно вносить правки в проект.

Основой причиной стало отсутствие жесткого промежуточного дедлайна на проверку.

Нашей командой было предложено следующей решение по графику в случае наличия подобных взаимопроверок:

- за 72 часа до дедлайна- выгрузка проекта, готового к оценке, на GitHub;
- за 48 часов- предоставление фидбека;
- за 24 часа- исправление выявленных недочетов.

2.4 Итоги этапа

К концу первой аттестации было полностью завершено предпроектное исследование и подготовлено ТЗ, которое было утверждено заказчиком. Это позволило перейти к этапу активной разработки с четким пониманием целей, требований и ограничений проекта. Все материалы документации стали основой для дальнейшей работы команды

3 Разработка и защита MVP

На текущем этапе (2 аттестация) необходимо создать работоспособное MVP приложение, которое будет демонстрировать основные возможности будущего продукта. Система должны выполнять ключевые сценарии работы: новые пользователи должны иметь возможность ознакомиться с приветственной страницей, оформить подписку(эмуляция системы оплаты), операторы должны управлять чатами, администраторы – настраивать интеграции и шаблонные ответы. А также должна быть возможность добавить следующие интеграции с сервисами: WhatsApp, VK, Email, Telegram.

Все эти функции будут реализованы в виде отдельных модулей в рамках backend-приложения по архитектуре MVC.

Общая схема архитектуры представлена на рисунке ниже.

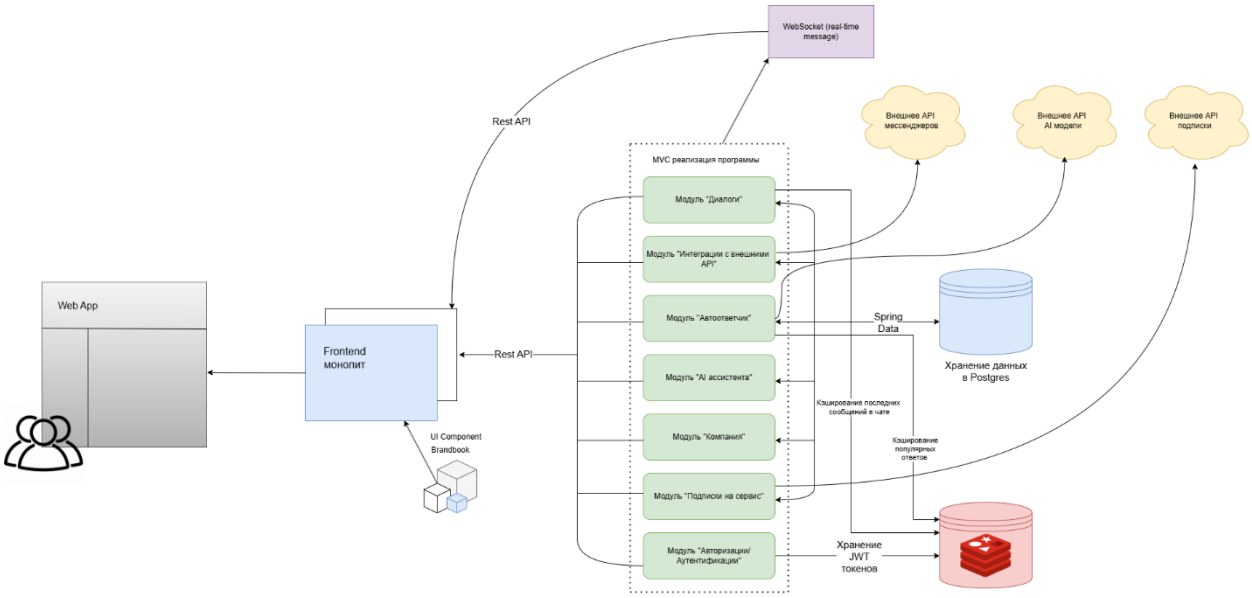

Рисунок 1 - Схема API

Пользователь взаимодействует с системой через веб-приложение, где клиентская часть реализована в виде фронтенд-монолита.

Фронтенд обменивается данными с серверной частью посредством REST API, а для передачи сообщений в реальном времени используется WebSocket. Серверная часть реализована по модели MVC и включает несколько функциональных модулей: модуль диалогов, модуль интеграции с внешними API, автответчик, AI-ассистент, модули компании и подписки, а также модуль авторизации и аутентификации.

Каждый из этих модулей выполняет свою задачу, обеспечивая раздельное управление логикой системы. Например, модуль "Диалоги" отвечает за обработку пользовательских сообщений, модуль AI-ассистента — за генерацию интеллектуальных ответов, а модуль интеграций — за взаимодействие с внешними API мессенджеров.

Все данные системы хранятся в реляционной базе данных PostgreSQL, к которой осуществляется доступ через Spring Data. Дополнительно используется кэширование на основе Redis: в нем сохраняются часто запрашиваемые данные, такие как популярные ответы и последние сообщения в чате.

3.1 Разработка Back-end

В качестве базы данных нами выбрана СУБД PostgreSQL. На данном этапе необходимо развернуть БД с тестовыми данными. Кроме того должны быть реализованы CRUD операции по основным сущностям проекта.

3.1.1 Модуль «Авторизация/аутентификация»

Данный модуль отвечает за регистрацию, вход и выход пользователей, а также проверку доступа к другим модулям (в зависимости от роли). Реализация будет использовать JWT-токены для авторизации, которые будут сохраняться в Redis для быстрого доступа и возможности инвалидации. Также необходимо обеспечить безопасную работу с паролями, для этого будет использоваться хэширование, https соединение, а также Spring Security.

3.1.2 Модуль «Подписки на сервис»

После аутентификации пользователь может оформить подписку на платные или условно-бесплатные функции системы. Этот модуль будет реализован в виде заглушки.

3.1.3 Модуль «Компания»

Этот модуль необходим для корпоративных пользователей. Он отвечает за управление данными компании, настройку корпоративных чатов, прав доступа сотрудников. Будет реализована CRUD-логика для работы с компанией.

3.1.4 Модуль «AI ассистента»

Один из ключевых модулей, предоставляющий интеллектуальные ответы в чате. Он будет взаимодействовать с внешним AI API (Qwen AI), обрабатывая сообщения пользователей и формируя осмысленные ответы. Модуль должен учитывать контекст шаблонного ответа и поддерживать кеширование в Redis для оптимизации скорости отклика.

3.1.5 Модуль «Автоответчик»

Инструмент, позволяющий загружать шаблонные ответы, на их основании будут формироваться AI-ответы. В случае невозможности ответа AI, будет выдаваться шаблонный ответ без изменений AI..

3.1.6 Модуль «Интеграции с внешними API»

Модуль интеграций — это ключевой компонент системы, обеспечивающий подключение внешних каналов связи к нашей платформе. Его основная задача — наладить двустороннюю коммуникацию между пользователями и клиентами через популярные мессенджеры и электронную почту, а также предоставить компаниям возможность внедрить чат на собственный сайт.

Для полноценной работы с электронной почтой система позволяет пользователю подключить свой email-аккаунт, указав адрес почты и пароль для IMAP-доступа. После подключения сообщения, поступающие на электронную почту, становятся доступны внутри системы, где пользователь может на них отвечать, как в обычном чате.

Поддержка мессенджеров реализуется через официальные API. Для интеграции с Telegram пользователь предоставляет токен своего бота и его имя — это позволяет системе принимать сообщения от пользователей Telegram и автоматически на них отвечать. Аналогичным образом организована интеграция с ВКонтакте: пользователь подключает сообщество, указав access token и название, после чего сообщения, поступающие в сообщество, обрабатываются нашей платформой.

Интеграция с WhatsApp реализуется через Business API. Для этого пользователь указывает API-токен и номер телефона, после чего становится возможен обмен сообщениями в рамках официальной схемы WhatsApp Business.

Помимо мессенджеров и почты, предусмотрена возможность встроить чат на сайт компании. Для этого система автоматически й системой генерируется уникальный API-ключ, который должен быть передан администратору сайта.

3.1.7 Модуль «Диалоги»

Основной модуль, обеспечивающий работу чата. Он будет хранить структуру диалогов, участников, историю сообщений. Реализуется поддержка WebSocket, позволяющая пользователям общаться в реальном времени. Также реализуется сохранение последних сообщений в Redis для ускоренного доступа и отображения истории чатов.

3.1.8 Связь модулей и взаимодействие

Frontend будет взаимодействовать с backend через REST API. Для чатов предусмотрен WebSocket, обеспечивающий реалтайм-обмен сообщениями. Коммуникация между внутренними модулями будет организована через сервисный слой.

3.2 Разработка Front-end

Предстоит создать систему интерфейсов, которая будет тесно интегрирована с бэкендом.

При открытии сайта пользователь попадает на информационную страницу, где он может ознакомиться с возможностями сайта, информацией о тарифах, а также зарегистрироваться или войти в профилью.

Точкой входа для новых пользователей станет страница регистрации.

Страница авторизации будет содержать форму входа. При успешном POST-запросе система перенаправит пользователя в его личный кабинет. Предусмотрим сценарий восстановления доступа.

После входа пользователь попадет на главную страницу.

Главная страница сайта – рабочая область**.** Она проектируется как единое пространство с тремя основными колонками: список чатов, область переписки и панель быстрых действий. В панели быстрых действий пользователь может перейти на страницу интеграций, подписки, шаблонных ответов, статистики, диалогов, настроек.

Для администраторов присутствует отдельный интерфейс управления компанией с возможностью добавлять сотрудников, настраивать права доступа и просматривать аналитику.

На странице интеграций визуализируем подключенные каналы (Telegram, Email и др.). Каждая интеграция будет активироваться через отдельные API.

На странице подписки пользователь может управлять подпиской.

На странице шаблонных ответов администратор может загрузить файл с шаблонными ответами, а также добавлять, редактировать и удалять шаблонные ответы.

На странице диалогов отображается список чатов, а также текущий чат, если таковой выбран.

На момент разработки MVP раздел статистики, является второстепенным.

Технически мы построим работу так: создадим макеты всех страниц в Figma, затем реализуем их на React, а после - поэтапно подключим к бэкенду, начиная с наиболее значимых для MVP страниц.

Наиболее приоритетными считаются страницы авторизации и регистрации, диалогов, главная страница, страница диалогов, страница интеграций, страница шаблонных ответов, приветственная страница.

4 Критерии успешности 2 этапа.

Успех второго этапа определяется созданием работоспособного MVP, которое демонстрирует ключевые сценарии использования: новый

пользователь регистрируется, авторизуется, ему доступна рабочая область с чатами, интеграциями и шаблонными ответами. Система должна работать с основными модулями — модулем авторизации, управления компанией, автоответчика, AI-ассистента, интеграций с email, Telegram, ВКонтакте и WhatsApp. Дополнительным показателем успеха является высокая скорость отклика (до 2с, при интернет соединении от 50мб/с) и надежность, подтвержденные тестированием(тесты покрывают более 70% кода), а также наличие актуальной документации и отчетов о выполнении задач.

Отчет подготовлен: Баранником Данилом Евгеньевичем
Дата: 13.04.2025